UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CONTROLE SUPERVISÓRIO MODULAR DE UMA CÉLULA DE MANUFATURA UTILIZANDO REDES DE PETRI INTERPRETADAS PARA CONTROLE

GUSTAVO DA SILVA VIANA

Rio de Janeiro, RJ - Brasil

Fevereiro de 2012

CONTROLE SUPERVISÓRIO MODULAR DE UMA CÉLULA DE MANUFATURA UTILIZANDO REDES DE PETRI INTERPRETADAS PARA CONTROLE

GUSTAVO DA SILVA VIANA

PROJETO SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA ESCOLA DE POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

Aprovado por:

JOÃO CARLOS DOS SANTOS BASILIO, Ph.D.
(Orientador)

MARCOS VICENTE DE BRITO MOREIRA , D.Sc.

SERGIO SAMI HAZAN, Ph.D.

Rio de Janeiro, RJ - Brasil

Fevereiro de 2012

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, porque sempre esteve ao meu lado me apoiando em todos os momentos e eternamente estará.

À minha mamãe Cláudia Márcia da Silva, vovó Julia Filomena da Silva e vovô José Cassimiro da Silva (já falecido) por me criarem, educarem e ensinarem que o conhecimento é um dos maiores bens que nós podemos ter. Tive muita sorte por Deus ter me colocado nessa família tão digna. Gostaria de agradecer ao meu tio Adeildo Antônio da Costa e também ao meu amigo Renato, por todas as caronas e conversas a caminho da faculdade.

Aos meus colegas de laboratório, Gustavo e Franscisco, sempre dispostos a ajudar. Aos meus colegas de classe, João, Flávio, Renan, Léo, Márcio, Bia, Isabela, Thiago Zaca, Rafael Jabour e tantos outros pela amizade, companheirismo e pelos *brainstorms* para poder passar nas matérias. Aos professores do laboratório, Lilian Kawakami, Marcos Moreira e Leonardo Bermeo, pela paciência, amizade e apoio técnico.

Por fim, gostaria de agradecer ao professor e orientador João Carlos dos Santos Basilio, que sempre me tratou como se fosse um filho, aconselhando, cobrando, orientando com leveza e carinho não de um chefe, mas de um pai. Seus ensinamentos serão levados por mim para a vida toda.

RESUMO

Gustavo da Silva Viana
UFRJ - Escola Politécnica

Projeto de Final de Curso
Fevereiro 2012

Controle Supervisório Modular de uma Célula de Manufatura utilizando Redes de Petri Interpretadas para Controle

Este trabalho consiste no projeto e implementação de uma célula de manufatura industrial, utilizando as teorias de controle supervisório modular e redes de Petri. A célula de manufatura é composta de uma esteira de fornecimento, uma esteira principal (onde a peça a ser manufaturada é pega pelo braço robótico e levada para a máquina correspondente), duas máquinas (aqui simuladas por esteiras) e um braço robótico. O controle do sistema é feito por meio de um controlador lógico programável (CLP) programado em linguagem Ladder, cujo código de programação foi obtido por meio da conversão de redes de Petri interpretada para controle em Ladder. Esta técnica, que está em crescente aperfeiçoamento, possibilita uma conversão direta do controle supervisório desenvolvido em redes de Petri para a linguagem Ladder utilizada na programação do supervisor no CLP.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A tecnologia moderna tem produzido, em escala crescente, sistemas com a finalidade de executar tarefas que, seja pela importância que possuem em seu contexto, seja por sua complexidade ou custo, justificam o esforço despendido na sua otimização e automação. Tais sistemas estão presentes em uma série de aplicações, incluindo por exemplo: a automação da manufatura, robótica, supervisão de tráfego, logística, sistemas operacionais, redes de comunicação de computadores, gerenciamento de bases de dados e otimização de processos distribuídos. Esses sistemas têm em comum a maneira pela qual percebem as ocorrências no ambiente à sua volta, o que se dá pela recepção de estímulos, denominados eventos. São exemplos de eventos o início e o término de uma tarefa e a percepção de uma mudança de estado em um sensor.

Os eventos são, por sua natureza, instantâneos e ocorrem em intervalos de tempo irregulares. Sistemas com estas características são denominados sistemas a eventos discretos (SED) [1], em oposição aos sistemas de variáveis contínuas, tratados pelas teorias de controle clássico e moderno. A natureza discreta dos SEDs faz com que os modelos matemáticos convencionais, baseados em equações diferenciais ou a diferenças, não sejam adequados para tratá-los. Por outro lado, a sua importância faz com que seja altamente desejável encontrar soluções para problemas relacionados ao seu controle. Em razão disso, existe uma intensa atividade de pesquisa voltada para a busca de modelos matemáticos adequados à sua representação. Dentre os métodos de modelar SEDs destacam-se as redes de Petri [2], [3], que definem graficamente a estrutura de um SED utilizando um grafo bipartido direcionado.

Através do modelo da planta e de suas especificações de funcionamento, é possível obter, pela teoria de controle supervisório [1], [4] e [5], um supervisor que garantirá que o sistema se comporte da maneira desejada. Considerando-se que os sistemas de manufatura são constituídos de subsistemas mais simples, surge também nesse contexto, as técnicas de controle modular [6], [7] e [8], que são muitos úteis para a construção de supervisores, de forma a controlá-los de maneira independente.

O trabalho aqui proposto utiliza os modelos de redes de Petri interpretada para controle (RPIC) para possibilitar a implementação do controle supervisório, em linguagem Ladder, utilizando o método proposto em [9]. O controle supervisório foi aplicado a uma planta real montada na bancada do Laboratório de Controle e Automação (UFRJ), diferentemente dos trabalhos [10] e [11], nos quais somente simulações foram realizadas. Uma importante contribuição deste trabalho é a aplicação do método desenvolvido em [9] para controle modular. O sistema a ser controlado neste projeto já fora utilizado em um trabalho anterior [12], porém empregando a modelagem por autômatos. Outras diferenças entre o sistema aqui considerado e aquele utilizado em [12] serão explicitadas no decorrer do texto.

Este trabalho está estruturado da seguinte maneira: no capítulo 2 são apresentados os conceitos básicos teóricos necessários para a compreensão do projeto, dentre os quais, redes de Petri, redes de Petri interpretadas para controle, linguagem Ladder e controle supervisório modular; no capítulo 3, há uma descrição dos componentes do sistema de manufatura, comportamento desejado para o mesmo e modelagem necessária para o controle; no capítulo 4 é abordada a implementação prática do sistema de manufatura, com diagrama de ligações e descrição de seu funcionamento; no capítulo 5, apresentam-se as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Sistemas a Eventos Discretos

Neste capítulo são apresentadas as bases da teoria de sistemas a eventos discretos necessárias para a elaboração deste projeto. Para tanto, este capítulo está estruturado da seguinte forma. Na seção 2.1, é apresentada a definição de sistemas a eventos discretos e alguns exemplos de SEDs. Na seção 2.2, aborda-se os fundamentos das redes de Petri, e na seção 2.3 considera-se as redes de Petri interpretadas para controle (RPIC). Na seção 2.4, há uma revisão de linguagem Ladder e na seção 2.5 é descrito como se faz a conversão de uma RPIC para Ladder, método de fundamental importância para o desenvolvimento deste trabalho. Por fim, na seção 2.6, é abordada a teoria de controle supervisório modular.

2.1 Modelagem de Sistemas a Eventos Discretos

De um modo geral, um sistema é uma parte limitada do universo que interage com o mundo externo através das fronteiras que o delimitam. Os sistemas a serem considerados neste trabalho são denominados sistemas a eventos discretos. Esses sistemas percebem as ocorrências no mundo externo através da recepção de estímulos, denominados eventos. São exemplos de eventos o início e o término de uma tarefa (mas não sua execução), a chegada de um cliente a uma fila ou a recepção de uma mensagem em um sistema de comunicação. A ocorrência de um evento causa, em geral, uma mudança interna no sistema, a qual pode ou não se manifestar a um observador externo. Além disso, uma mudança pode ser causada pela ocorrência de um evento interno ao próprio sistema, tal como o término de uma atividade ou de uma temporização. Em qualquer caso, essas

mudanças se caracterizam por serem abruptas e instantâneas; ao perceber um evento, o sistema reage imediatamente, acomodando-se em tempo nulo a uma nova situação, onde permanece até que ocorra um novo evento. Desta forma, a simples passagem do tempo não é suficiente para garantir que o sistema evolua; para tanto, é necessário que ocorram eventos, sejam esses internos ou externos. Note ainda que a ocorrência desses eventos pode depender de fatores alheios ao sistema, que não podem, em geral, ser previstos pelo sistema. Assim, pode-se definir Sistemas a Eventos Discretos (SEDs) como um sistema dinâmico que evolui de acordo com a ocorrência abrupta de eventos físicos, em intervalos de tempo em geral irregulares e desconhecidos.

O conjunto de eventos discretos de um SED, pode ser considerado como o alfabeto do sistema. Seguindo esse raciocínio, cada sequência possível de eventos de comprimento finito é chamada de palavra, e o conjunto de todas as sequências possíveis de comprimento finito é chamada de linguagem. Linguagens são importantes para acompanhar a evolução dos estados de um SED, uma vez que estes não podem ser modelados por equações diferenciais como os sistemas dinâmicos de variáveis contínuas. Apesar das linguagens representarem uma maneira conveniente de modelar SEDs, na maioria dos casos de interesse prático, as linguagens são complexas e não podem ser enumeradas. Para representá-las, duas estruturas são comumente utilizadas: o autômato e a rede de Petri. Este trabalho utilizará modelos por redes de Petri.

Uma breve revisão da teoria de redes de Petri será apresentada a seguir.

2.2 Fundamentos Básicos das Redes de Petri

Uma rede de Petri é um grafo orientado bipartido [2], [3], ou seja, possui dois tipos de vértices que nunca são ligados entre si. Em outras palavras, uma rede de Petri é um dispositivo que manipula eventos respeitando regras bem definidas. Ela possibilita modelar e visualizar comportamentos como paralelismo, concorrência, sincronização e recursos compartilhados [2].

2.2.1 Grafo de uma Rede de Petri

O grafo de uma rede de Petri é composto, basicamente, de lugares, transições e arcos, aos quais são atribuídos pesos. Os lugares são representados por círculos, e contêm as informações relacionadas ao estado e condições do sistema. As transições são representadas por barras, e estão associadas aos eventos ocorrentes. Os arcos, por fim, são representados por setas, e determinam as relações entre as transições e os lugares.

Definição 2.1 *(grafo de uma rede de Petri) Um grafo de uma rede de Petri é uma quádrupla* (P, T, A, ω)*, em que:*

- P=$\{p_1,p_2,p_3,...,\ p_n\}$ *é o conjunto finito de* lugares*;*
- T=$\{t_1,t_2,t_3,...,\ t_m\}$ *é o conjunto finito de* transições*;*
- $A \subseteq (P \times T) \cup (T \times P)$ *é o conjunto dos arcos que ligam lugares a transições e transições a lugares;*
- ω*:* $A \rightarrow \{1,2,3...\}$ *é a função de ponderação dos arcos.*

A figura 2.1 mostra um exemplo básico de um grafo de rede de Petri, cujos componentes são P=$\{p_1,p_2\}$, T=$\{t_1\}$, A=$\{(p_1,t_1),(t_1,p_2)\}$, $\omega(p_1,t_1)$=2 e $\omega(t_1,p_2)$=1.

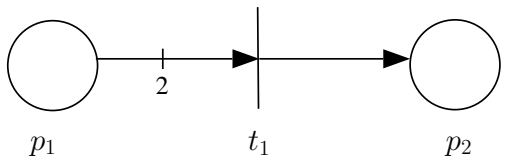


Figura 2.1: Exemplo de grafo de uma rede de Petri

2.2.2 Redes de Petri

A definição de grafo de rede de Petri não é suficiente para modelar o comportamento dinâmico dos sistemas em análise. Faz-se necessário um elemento que indique o estado do sistema e quais transições estão habilitadas, para que, ao ocorrer o evento associado a essas transições, haja a indicação da mudança do estado do sistema (chamado de "disparo").

Para solucionar esse problema, são utilizadas fichas, também referidas como *tokens*. Elas são atribuídas aos lugares e representam as condições e a evolução de uma rede de Petri. O estado atual é retratado pela distribuição das fichas nos lugares. Isso leva ao conceito de rede de Petri com marcação.

Definição 2.2 *(rede de Petri com marcação) Uma rede de Petri com marcação é uma quíntupla* $(P, T, A, \omega, \underline{x})$*, em que:*

- *(P,T,A,ω) é um grafo de rede de Petri;*
- $\underline{x} = \left[\begin{array}{cccc} x(p_1) & x(p_2) & \ldots & x(p_n) \end{array} \right]^T$ *é um vetor coluna que indica a marcação dos lugares da rede de Petri, ou seja, o número de fichas em cada lugar.*

A figura 2.2 mostra a rede de Petri da figura 2.1 com duas marcações diferentes: $\underline{x} = \left[\begin{array}{cc} 1 & 0 \end{array} \right]^T$ (figura 2.2(a)) e $\underline{x} = \left[\begin{array}{cc} 4 & 1 \end{array} \right]^T$ (figura 2.2(b)).

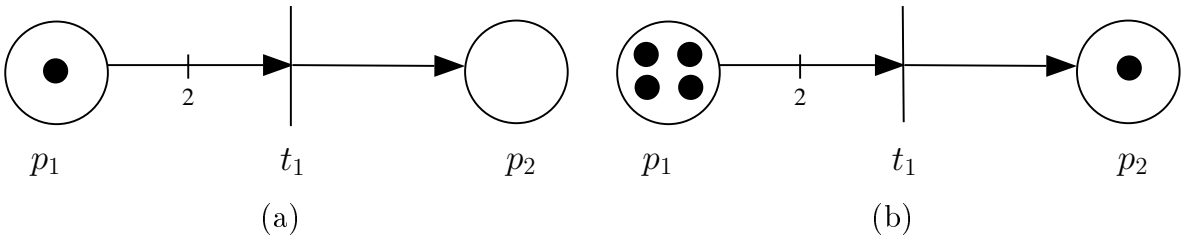


Figura 2.2: Exemplos de redes de Petri com marcações.

Um conceito importante relacionado à marcação de uma rede de Petri é o de rede de Petri segura, que é apresentado na definição a seguir.

Definição 2.3 *(rede de Petri segura) Uma rede de Petri é dita* ***segura*** *se todos os seus lugares possuírem, a qualquer momento, no máximo uma ficha, isto é, não há a possibilidade de mais de uma ficha por lugar.*

Todas as redes de Petri consideradas neste trabalho são seguras e a justificativa para isso será apresentada mais à frente.

2.2.3 Dinâmica das Redes de Petri

O mecanismo de transição de estados em uma rede de Petri é possibilitado pelo movimento dos *tokens*(fichas) através da rede. Quando uma transição está habilitada, dizemos que ela pode disparar ou que pode ocorrer (o termo "disparo" é padrão na literatura de rede de Petri). Uma transição só está habilitada quando cada um de seus lugares de entrada tem um número de fichas igual ou maior que o peso do arco que o liga à transição. Tal afirmação é elucidada pela equação 2.1.

Seja $I(t_j)$ o conjunto de lugares de entrada da transição t_j. Então a transição t_j está habilitada se:

$$x(p_i) \geq \omega(p_i, t_j), \forall p_i \in I(t_j) \tag{2.1}$$

Na figura 2.3, a transição t_1 não está habilitada, pois o lugar p_1 só possui 1 ficha enquanto o peso do arco $\omega(p_1, t_1)$ é igual a 2.

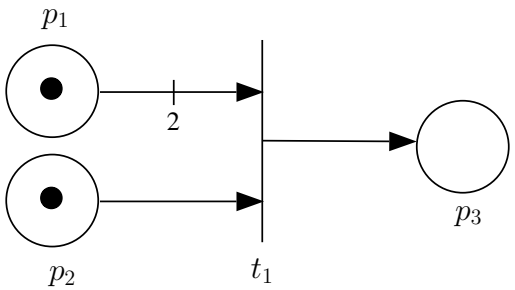


Figura 2.3: Transição não habilitada.

Por outro lado, a figura 2.4 mostra uma situação em que a transição t_1 está habilitada, uma vez que cada lugar de entrada possui um número de fichas igual ao peso do arco que o liga à transição. Com isso, o disparo pode ocorrer.

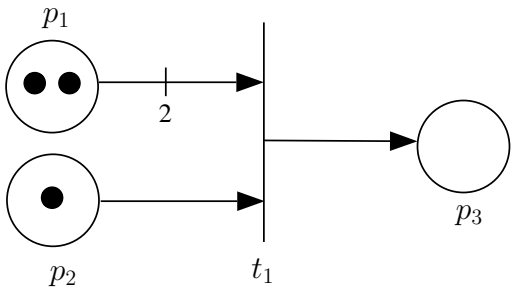


Figura 2.4: Transição habilitada.

Quando uma transição dispara, a distribuição das fichas é normalmente alterada, indicando a mudança nas condições do sistema. Cada lugar de entrada da transição que foi disparada perde uma quantidade de fichas equivalente ao peso do arco que o liga à transição, e cada lugar de saída da transição disparada ganha, da mesma forma, uma quantidade de fichas equivalente ao peso do arco que liga a transição disparada a esse lugar. A figura 2.5 ilustra o processo de disparo de uma transição.

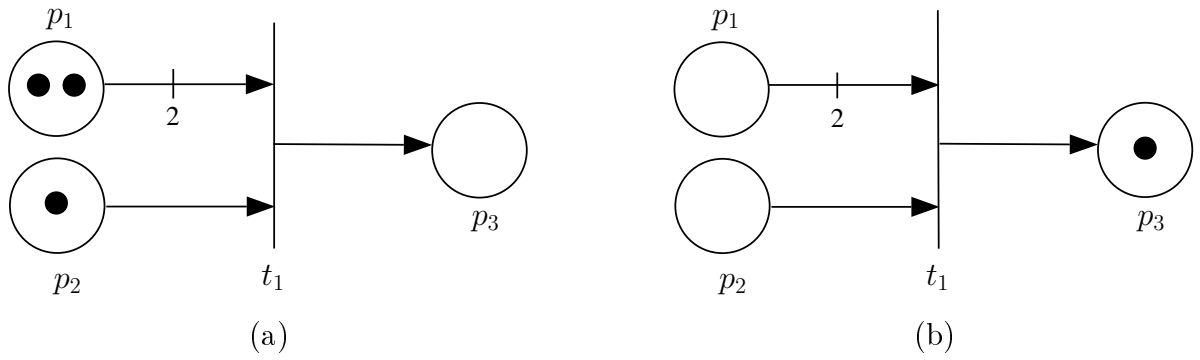


Figura 2.5: Rede antes do disparo (a) e rede depois do disparo (b).

Na figura 2.5(a), a transição t_1 está habilitada. Quando t_1 dispara, os lugares de entrada p_1 e p_2 perdem a quantidade de fichas correspondentes ao peso de seus respectivos arcos ligados a transição t_1, enquanto o lugar p_3 recebe a quantidade de fichas correspondente ao peso do arco que o liga a transição t_1. A situação da rede, após o disparo da transição t_1 é mostrada na figura 2.5(b).

2.2.4 Rede de Petri Rotulada

As redes de Petri que possuem suas transições associadas a eventos são chamadas de redes de Petri rotuladas. A representação dessas redes é muito semelhante àquela vista anteriormente neste capítulo, com uma diferença: junto às transições posicionam-se os eventos que devem ocorrer para que a transição dispare.

Definição 2.4 *(rede de Petri rotulada) Uma rede de Petri rotulada é uma séptupla*

$$R = (P, T, A, \omega, \underline{x}, E, l),$$

em que:

- *(P,T,A,ω,$\underline{x}$) é uma rede de Petri com marcação;*
- *E é o conjunto dos eventos associados às transições;*
- *$l : T \rightarrow E$ é a função de rotulação das transições.*

É importante salientar que a função l não é necessariamente injetora, isto é, é possível haver um mesmo evento associado a mais de uma transição. A figura 2.6 mostra uma rede muito simples, que ilustra o conceito de rede de Petri rotulada. Na figura 2.6, o lugar

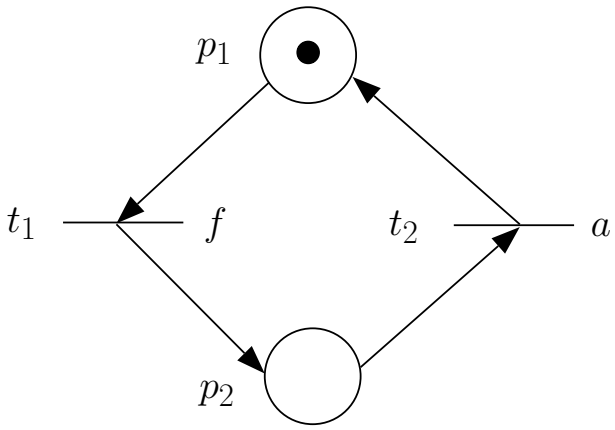


Figura 2.6: Rede de Petri rotulada.

p_1 representa o estado "porta aberta", enquanto p_2 representa o estado "porta fechada". Apesar da transição t_1 estar habilitada, ela só irá disparar e mudar para o estado "porta fechada" quando o evento f (fechar a porta) ocorrer. Analogamente acontece quando a porta está fechada. Só ocorre o disparo da transição t_2 no momento que o evento a (abrir a porta) ocorrer.

2.2.5 Conflitos

Um conflito ocorre quando duas ou mais transições têm pelo menos um lugar de entrada em comum, e "disputam" a ficha (ou fichas) desse lugar (ou lugares) para que possam disparar. Existem três tipos de conflito: estrutural, efetivo e real.

Conflito estrutural é aquele que leva em conta apenas a estrutura da rede de Petri, independente do estado do sistema. Um conflito estrutural representa a possibilidade de ocorrência da disputa por fichas, caso certa distribuição de fichas seja atendida. A figura

2.7 ilustra esse conceito. Nela há dois conflitos, $(p_1, \{t_1, t_2\})$ que indica uma possível disputa das transições t_1 e t_2 por uma ficha em p_1 e $(p_2, \{t_2, t_3, t_4\})$ que indica uma possível disputa de t_2, t_3 e t_4 por uma ficha em p_2.

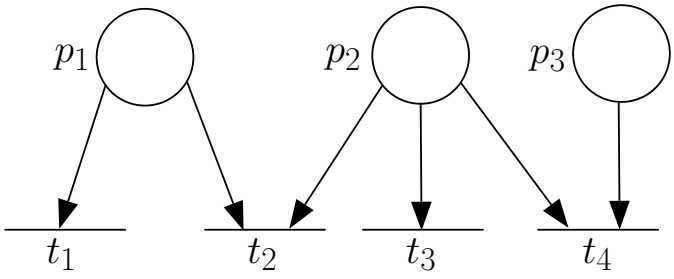


Figura 2.7: Exemplo de grafo com conflito estrutural.

Conflito efetivo é aquele que leva em conta, além da estrutura da rede de Petri, a distribuição de fichas no momento analisado, ou seja, indica um estado de conflito em um determinado momento, para determinada marcação. Enquanto o conflito estrutural é permanente, o conflito efetivo tem uma duração, que depende da dinâmica da rede de Petri. Todo conflito efetivo depende da existência de um conflito estrutural, mas a recíproca não é verdadeira. A figura 2.8 apresenta o grafo de rede de Petri da figura 2.7 com a marcação $\underline{x}_0 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$. A marcação não interfere nos conflitos estruturais, que continuam os mesmos. Contudo, os conflitos efetivos são $(p_1, \{t_1, t_2\}, \underline{x}_0)$ e $(p_2, \{t_2, t_3\}, \underline{x}_0)$. O conflito estrutural $(p_2, \{t_2, t_3, t_4\})$ não representa um conflito efetivo porque, para o estado $\underline{x}_0$, t_4 não está habilitada, logo, não disputa a ficha de p_2 com t_2 e t_3. Por isso, a disputa pela ficha de p_2 é representada pelo conflito $(p_1, \{t_2, t_3\}, \underline{x}_0)$. Caso p_3 recebesse uma ficha, ou seja, para o estado $\underline{x}_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, t_4 estaria habilitada, e o conflito efetivo $(p_2, \{t_2, t_3\}, \underline{x}_1)$ seria substituído por $(p_2, \{t_2, t_3, t_4\}, \underline{x}_1)$.

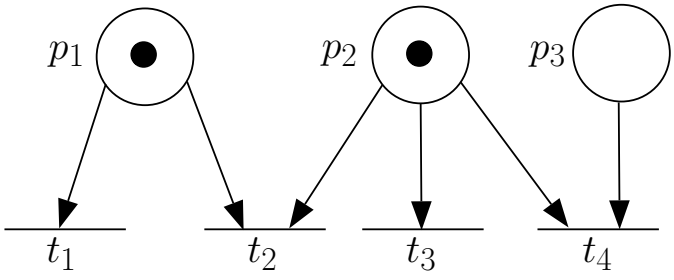


Figura 2.8: Exemplo de grafo com conflito efetivo.

Conflito real é o conflito que, além de considerar a marcação da rede de Petri, considera também os eventos e condições associados às transições, levando-se em conta se há ou não a chance das transições em um conflito efetivo dispararem simultaneamente. Dessa forma, um conflito só é considerado real caso haja realmente a possibilidade de disputa por fichas no momento, e os eventos e condições associados às transições envolvidas não excluirem essa possibilidade.

2.3 Rede de Petri Interpretada para Controle

2.3.1 Definição

As redes de Petri interpretadas para controle (RPIC) representam o controlador de um SED. Nelas, uma transição habilitada só é disparada quando um evento ocorre e se certas condições predeterminadas forem verdadeiras no momento da ocorrência do evento. Isso quer dizer que além, obviamente, das condições relacionadas à distribuição das fichas, que determina se uma transição está habilitada ou não e da associação de transições com eventos proposta pela rede de Petri rotulada, a RPIC considera condições externas à rede. Além disso, os lugares passam a ter uma função além da representação das condições do sistema. Quando uma ficha é introduzida em um lugar, alguma ação ou operação relacionada a esse lugar é efetuada. É importante lembrar que os lugares continuam tendo, nas RPICs, a função de indicar as condições para o disparo de uma transição. Assim, independentemente das condições associadas a uma transição serem válidas, e do evento associado a essa mesma transição ocorrer, o disparo da transição continua dependendo, primeiramente, das condições indicadas pela distribuição das fichas na rede de Petri.

As RPICs aqui definidas são também temporizadas, ou seja, as transições podem ser associadas a atrasos de disparo. Dessa forma, o conjunto de transições T pode ser particionado como $T = T_0 \dot{\cup} T_D$, em que T_0 denota o conjunto de transições não temporizadas e T_D o conjunto de transições temporizadas, ou seja, transições que apresentam atraso no disparo. As transições temporizadas são identificadas pelo atraso d_j indicado ao lado delas.

De posse dos elementos acima, podemos apresentar a seguinte definição.

Definição 2.5 *(RPIC) Uma rede de Petri interpretada para controle é uma décupla:*

$$R_{pic} = (P, T, A, \omega, \underline{x}, E \cup e, C, D, O, Q)$$

em que:

- *(P,T,A,ω,$\underline{x}$) é uma rede de Petri marcada segura;*
- $T = T_0 \dot{\cup} T_D$*;*
- $E = \{e_j : t_j \in T_0\}$ *é o conjunto de eventos;*
- *e é o evento que sempre ocorre;*
- $C = \{c_j : t_j \in T_0\}$ *é o conjunto de condições;*
- $D = \{d_j \in \mathbb{R} : t_j \in T_D\}$ *é o conjunto de atrasos de disparo associados às transições temporizadas de* T_D*;*
- *O é o conjunto de operações associadas aos lugares;*
- *Q é o conjunto de ações booleanas ou impulsionais associadas aos lugares.*

Com relação aos elementos da RPIC, percebe-se que as transições temporizadas não apresentam associações a eventos ou condições. Assim se uma transição $t_j \in T_D$ estiver habilitada, de acordo com as circunstâncias apresentadas na subseção 2.2.3, ela irá disparar exatamente após o tempo de seu atraso d_j. Por outro lado, uma transição não temporizada tem sempre uma condição e evento associado a ela e disparará apenas se estiver habilitada. O evento "e" está associado a transições que sempre ocorrem, e sendo assim, sua condição sempre possui valor 1, isto é, verdadeiro. Sempre que não houver um evento associado a alguma transição explicitamente num grafo de RPIC, está implícito que ela está associada a um evento "e". As ações booleanas são ações que permanecem sendo executadas apenas enquanto o lugar correspondente tiver uma marcação; por exemplo, uma lâmpada permanece acesa enquanto o lugar associado ao acendimento da lâmpada possuir uma ficha, e será apagada quando esse lugar perder a ficha. Note que, nesse caso, não é necessário ter um outro lugar com uma ação associada ao desligamento da lâmpada.

Uma ação impulsional, por outro lado, pressupõe a existência de um dispositivo externo à rede de Petri que memorize o comando enviado para o sistema. Por exemplo, a marcação de um lugar pode indicar o ligamento de uma chave on/off, que permanecerá ligada mesmo que o lugar equivalente se desmarque em seguida. Essa chave só será desligada quando outro lugar, que esteja associado ao desligamento dessa chave, for marcado. Uma operação é uma ação que executa algum tipo de alteração no valor de uma variável, como, por exemplo, em um contador.

A figura 2.9 representa graficamente a interação do controlador a eventos discretos descrito pela RPIC com o ambiente externo (sistema controlado e operador) e uma parte de processamento de dados. As setas indicam o sentido do fluxo de sinais e de dados. A RPIC recebe informações a respeito das condições do ambiente c_j^e e os eventos externos e_j, e envia sinais de saída para o ambiente, que são as ações q_i. O controlador a eventos discretos envia também ordens de operação o_i para o sistema de processamento de dados e recebe informações a respeito de condições internas associadas aos dados processados c_j^p.

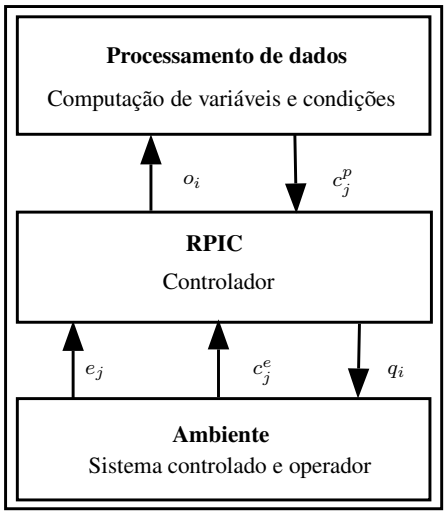


Figura 2.9: Diagrama representando o fluxo de sinais e dados em um sistema de automação descrito por uma RPIC.

A representação das ações e eventos em uma RPIC é apresentada na figura 2.10. Ao lado do lugar p_i, encontram-se a operação o_i e a ação q_i, que são executadas quando o lugar p_i recebe uma ficha. Ao lado da transição t_j localizam-se o evento e_j e a condição

$c_j = c_j^e c_j^p$. É importante ressaltar que se t_j fosse uma transição temporizada, então no lugar do evento e_j e da condição c_j estaria associado o atraso de disparo d_j .

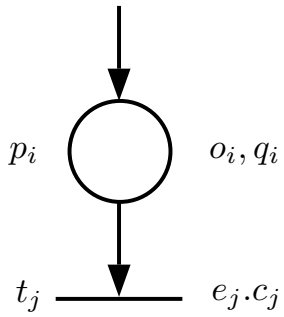


Figura 2.10: Representação das ações e operações associadas aos lugares e das condições e eventos associados às transições em uma RPIC.

Observação 2.1 *De acordo com o que foi discutido anteriormente, o simples fato de haver uma ficha em um lugar caracteriza a ação associada a esse lugar. Dessa forma é desnecessário definir um lugar que possa ter mais de uma ficha, justificando assim a adoção de redes de Petri seguras. No caso da modelagem necessitar de lugares com mais de uma ficha, a consideração é feita na parte de processamento de dados.*

Para ilustrar os conceitos apresentados nessa subseção, apresentaremos um exemplo de RPIC.

Exemplo 2.1 *A figura 2.11 mostra a RPIC de um sinal de trânsito. O sinal verde tem duração de dois minutos, tempo após o qual a luz amarela é acesa durante cinco segundos. Depois da amarela, a luz vermelha é acionada, obrigando a parada dos carros por um minuto. Há um botão para que o pedestre possa forçar a mudança do sinal verde para o amarelo, porém esse evento só pode ocorrer na parte da noite.*

As três ações executadas nesse sistema são ações booleanas. As lâmpadas verde, amarela e vermelha só permanecem ligadas enquanto os lugares p_1, p_2 e p_3, respectivamente, estiverem marcados. As transições t_1, t_2 e t_3 são temporizadas, com atrasos d_1 = 2 minutos, d_2 = 5 segundos e d_3 = 1 minuto. Quando o lugar p_1 estiver marcado, a transição t_1 estará habilitada. Seu disparo, contudo, só ocorrerá após o atraso d_1, que é

o tempo de duração do sinal verde. Quando ocorrer esse disparo, a ficha sai de p_1*, apagando a luz verde, e marca* p_2*, acendendo a luz amarela. Neste momento, a transição* t_2 *se habilita, e dispara após os cinco segundos equivalentes ao sinal amarelo. Esse disparo desmarca* p_2*, apagando a luz amarela, e marca* p_3*, acendendo a luz vermelha e habilitando* t_3*. Após o atraso* d_3*, a transição dispara, e o sinal volta a ficar verde. Quando o sinal estiver verde na parte da noite, o evento "apertar botão" pode ocorrer. Isto fará com que instantaneamente o sinal mude de verde para amarelo.*

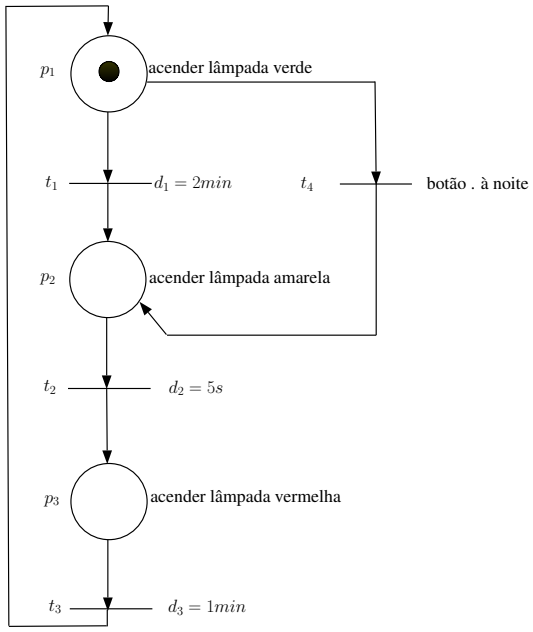


Figura 2.11: Sinal de trânsito com botão para pedestre.

Nesse sistema, há um conflito real. Caso o botão seja apertado com a condição "noite" sendo satisfeita, no exato instante em que os 2 minutos de sinal verde se esgotam, as transições t_1 *e* t_4 *disparam simultaneamente. Contudo, o lugar* p_1 *possui apenas uma ficha. Logo, as duas transições estão disputando a única ficha de* p_1*.*

Na subseção a seguir será apresentada uma solução para esse problema.

2.3.2 RPIC com Prioridades

A rede de Petri interpretada para controle com prioridades (RPICP) surge para eliminar dúvidas relacionadas a conflitos. Assim, caso duas ou mais transições estejam disputando

fichas, podendo disparar ao mesmo momento, a prioridade definida na RPICP indicará qual das transições terá seu disparo efetuado.

Definição 2.6 *(RPICP) Uma rede de Petri interpretada para controle com prioridades é uma dupla* (N, P_r)*, em que* N *é uma RPIC e* $P_r = \{T_{p1}, T_{p2}, \ldots, T_{pl}\}$*, sendo* l *o número de conflitos efetivos em* N *e* T_{pj} *uma sequência de transições envolvidas em um conflito efetivo, organizadas em ordem decrescente de prioridades para o disparo.*

A figura 2.12 mosta um conflito efetivo, em que t_2 possui prioridade em relação a t_1.

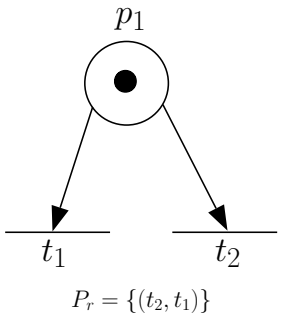


Figura 2.12: RPIC com prioridade.

2.4 A Linguagem Ladder

A linguagem Ladder, diagrama Ladder, ou ainda diagrama de escada é um linguagem gráfica para a programação de controladores lógicos programáveis (CLPs) no qual as funções lógicas são representadas através de contatos e bobinas, de modo análogo a um esquema elétrico com os contatos dos transdutores e atuadores. A linguagem Ladder está entre as cinco linguagens de programação de CLPs definidas pela IEC 1131-3 [13], quais sejam FBD (Function block diagram), LD (Ladder diagram), ST (Structured text), IL (Instruction list) e SFC (Sequential function chart). O nome Ladder (escada em inglês) provém da disposição dos contatos e bobinas, que de maneira geral, estão na horizontal, lembrando o formato de uma escada.

O diagrama Ladder é uma linguagem simbólica que utiliza diversos componentes, tais como: contatos, bobinas, temporizadores, contadores, instruções de comparação, instruções de cálculos matemáticos elementares e instruções de cálculos matemáticos complexos. Neste trabalho nos limitaremos à utilização de somente alguns desses componentes citados, exemplificados por elementos do manual do CLP s7-1200 Siemens [14] utilizado na implementação prática.

2.4.1 Contatos NA e NF

Os contatos são componentes fundamentais de um diagrama Ladder. A CPU executa esta instrução verificando o valor do bit endereçado. Os contatos podem ser normalmente abertos (NA) ou normalmente fechados (NF).

Nos contatos NA, se o bit estiver no estado lógico 0, ela retorna um valor lógico falso e portanto, não dá continuidade lógica no trecho Ladder em que a instrução está inserida. Se o bit endereçado estiver no valor lógico 1, a instrução retorna um valor lógico verdadeiro, dando assim continuidade lógica ao trecho em que está inserida. A figura 2.13 representa o contato NA nomeado "IN", em linguagem Ladder, associado à variável booleana $\%I0.0$.

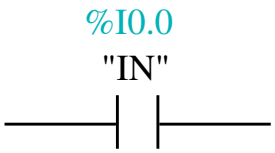


Figura 2.13: Contato normalmente aberto.

O contato normalmente fechado (NF), simbolizado na figura 2.14, trabalha de forma inversa ao NA. Quando o estado lógico do bit for 1, ele retorna um valor "falso", e se o estado for 0, ele retorna valor "verdadeiro".

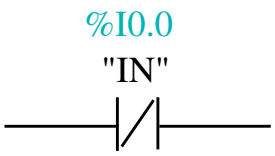


Figura 2.14: Contato normalmente fechado.

2.4.2 Contatos “tipo P” e “tipo N”

O contato “Scan Positive Signal Edge and Operand” é um tipo de contato que funciona de maneira análoga ao contato NA, porém ao invés de fechar quando a variável associada apresenta valor lógico igual a 1, esse contato fecha somente durante o ciclo de varredura imediatamente após a subida da variável booleana associada, ou seja, da sua passagem de 0 para 1. No próximo ciclo de varredura após a subida da variável booleana associada, o contato abre, pois nenhuma mudança positiva de estado lógico foi detectada.

Como mostra a figura 2.15, há duas variáveis associadas ao contato: a primeira, variável “IN” posicionada acima do contato, é a variável cuja mudança positiva de estado lógico se deseja detectar, e a segunda variável, “*mbit*”, posicionada abaixo do contato, é a variável que armazena o valor da primeira variável durante o último ciclo de varredura do programa do controlador. O contato tipo P detecta a subida da primeira variável associada comparando-a ao valor da segunda variável associada, ou seja, o contato compara seu valor atual com o seu valor durante a última varredura. A partir dessa comparação, caso a subida do valor lógico seja detectada, o contato fecha e, caso não haja variação no valor da variável, o contato abre.

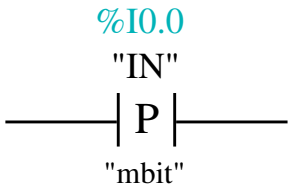


Figura 2.15: Contato tipo P.

O contato “Scan Negative Signal Edge and Operand” (ou contato tipo N), representado na figura 2.16, é um tipo de contato utilizado para detectar a descida de uma determinada variável booleana. Este contato funciona de maneira análoga ao contato tipo P, só que, ao invés de fechar na subida lógica da variável associada, esse contato fecha na descida da variável booleana associada, ou seja, da sua passagem de 1 para 0.

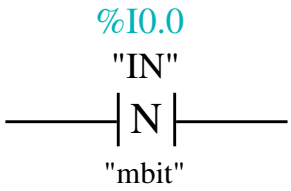


Figura 2.16: Contato tipo N.

2.4.3 Bobinas

Assim como os contatos, as bobinas são componentes básicos de um diagrama Ladder. Elas atualizam as informações de saída modificando o estado lógico de variáveis booleanas de acordo com o seu tipo. Seus principais tipos são: bobina simples, bobina SET e bobina RESET.

Caso a lógica que a antecede seja verdadeira, a bobina simples é "energizada", isto é muda seu valor lógico para "verdadeiro" assim como o da variável associada a ela. Caso a lógica anterior à bobina simples se torne falsa, ela é imediatamente "desenergizada", logo modificando seu valor lógico para "falso". A figura 2.17 ilustra uma bobina simples.

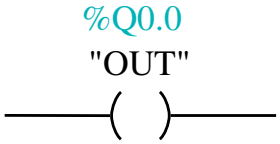


Figura 2.17: Bobina simples.

As bobinas SET e RESET funcionam de uma forma diferente. Caso a lógica que antecede a bobina SET seja verdadeira, o valor da variável será levado para 1. Mesmo que após isso a lógica se torne falsa, o valor permanece sendo 1. Para que o valor da variável seja levado para 0, é necessário que em alguma outra linha do diagrama exista uma bobina de RESET associada à mesma variável booleana e que esta bobina seja energizada. A representação da bobina SET é mostrada na figura 2.18 e da bobina RESET na figura 2.19. Note que ambas as bobinas estão associadas à mesma variável.

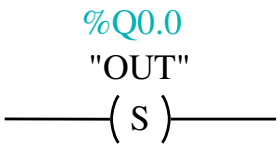


Figura 2.18: Bobina SET.

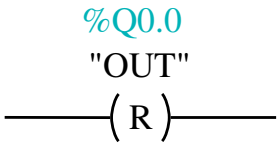


Figura 2.19: Bobina RESET.

A bobina RESET funciona de forma inversa à bobina SET. Ela atualiza e retém o valor lógico da variável associada para "falso". O valor dessa variável se tornará "verdadeiro" caso uma bobina SET em alguma outra linha do diagrama, associada a essa mesma variável, seja energizada.

2.4.4 Temporizador TON

Os temporizadores são componentes usados para criar atrasos pré-programados. Dentre os tipos de temporizadores, o tipo utilizado neste trabalho é o TON (Time On Delay). Um bloco temporizador TON adiciona um atraso, em milissegundos, igual ao valor pré-configurado dentro do bloco, à condição lógica antecedente. A figura 2.20 mostra um bloco TON, de onde se pode perceber que estão associadas ao bloco TON, quatro variáveis. Cada variável e sua respectiva descrição são mostradas na tabela 2.1.

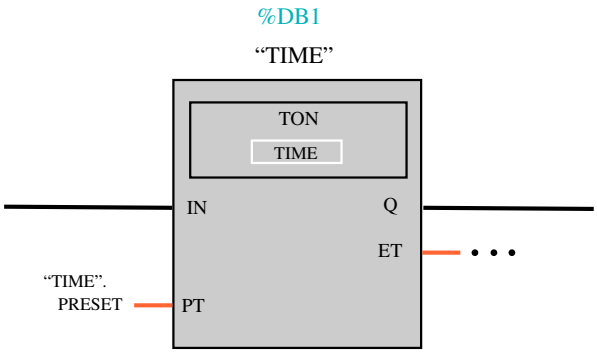


Figura 2.20: Temporizador TON.

Tabela 2.1: Parâmetros associados a um bloco temporizador do tipo TON de um controlador SIEMENS.

Parâmetros	Descrição
IN	sinal lógico de entrada
Q	sinal lógico de saída
PT	tempo de atraso
ET	tempo percorrido

Se IN tornar-se falso antes que o tempo percorrido ET alcance o valor armazenado PT, ET é zerado e só se contará o tempo novamente quando o valor de IN for verdadeiro. Isso ocorre, porque o temporizador do tipo TON não é retentivo, isto é, não retém o valor do parâmetro ET em caso de descida do valor lógico da entrada IN.

2.4.5 Blocos Comparadores

Os blocos comparadores são muito úteis na lógica dos diagramas Ladder. Eles comparam dois valores de mesmo tipo de dado. Por exemplo: inteiro com inteiro, real com real, booleano com booleano (digital). Quando a comparação do contato for verdadeira, ele será então ativado. A figura 2.21, mostra a comparação entre dois valores inteiros: "IN1", endereçado no bit %IW64 e 30000.

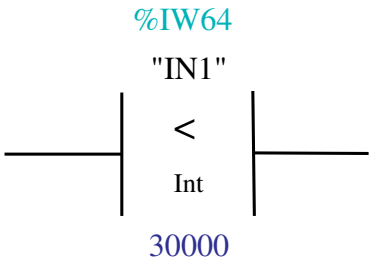


Figura 2.21: Bloco comparador "menor que".

Além de poder comparar dois valores, é possível também comparar três valores de modo a saber se um deles encontra-se dentro ou fora da faixa limitada entre os outros dois. Esses blocos são comparadores tipo "range". O bloco "IN_RANGE" verifica se um valor encontra-se dentro da faixa e o "OUT_RANGE" fora. A figura 2.22 ilustra um

bloco “IN_RANGE”, em que a variável “IN” (endereçada no bit %IW64) tem como valor mínimo 0 e valor máximo 15000. Caso ela esteja na faixa, a saída do bloco terá o valor lógico “verdadeiro”.

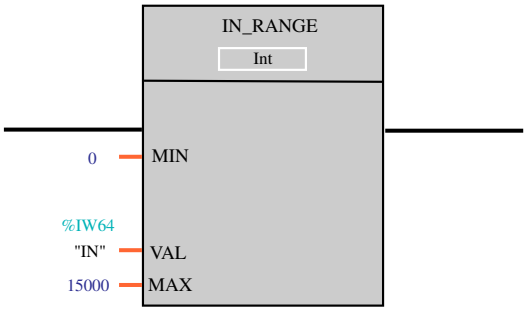


Figura 2.22: Bloco “IN_RANGE”.

2.5 Conversão de RPICs para Ladder

Após as breves revisões de RPICs e de diagrama Ladder apresentadas, respectivamente, nas seções 2.3 e 2.4, podemos considerar agora o método de conversão RPIC para Ladder proposto em [9]. O método consiste em construir um diagrama Ladder composto de 5 módulos: módulo de inicialização, módulo de eventos, módulo de condições para o disparo da transição, módulo da dinâmica e módulo das ações.

Para explicarmos o método de conversão, utilizaremos o sistema de sinal de trânsito com botão de pedestre, descrito na seção 2.3. Este sistema é ilustrado na figura 2.23 e a correspondente RPIC está representada na figura 2.11. Para resolver o conflito existente entre as transições t_1 e t_4, vamos considerar que t_4 tem prioridade sobre t_1, isto é, $P_r = \{(t_4, t_1)\}$.

A partir da RPIC da figura 2.11, pode-se construir o diagrama Ladder baseado no método [9]. Primeiramente, construiremos o módulo de inicialização.

2.5.1 Módulo de Inicialização

O módulo de inicialização está associado ao estado inicial da rede de Petri, isto é, ele distribui as fichas para os lugares apropriados definindo a marcação inicial da rede. O

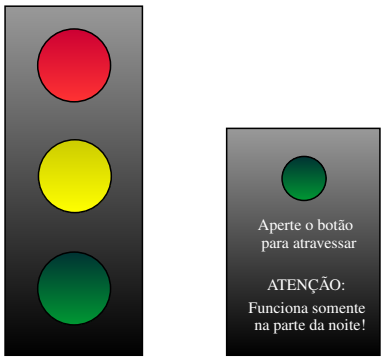


Figura 2.23: Sinal de trânsito com botão de pedestre.

módulo de inicialização possui um contato normalmente fechado associado a uma variável binária interna, energizando logicamente bobinas associadas a lugares marcados no primeiro ciclo de varredura. Após o primeiro ciclo de varredura o contato normalmente fechado é aberto, não permitindo que esta linha seja executada novamente. A figura 2.24 mostra o módulo de inicialização da rede de Petri ilustrada na figura 2.11.

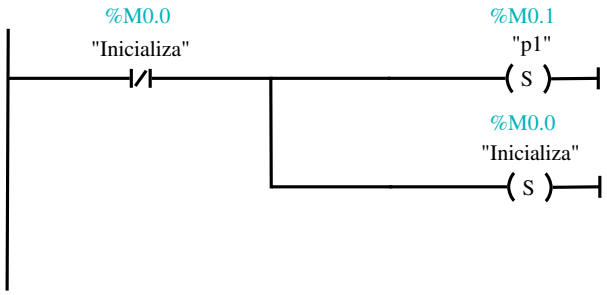


Figura 2.24: Módulo de inicialização de um sinal de trânsito com botão de pedestre.

2.5.2 Módulo de Eventos

O módulo de eventos consiste na interpretação de eventos externos como eventos internos, gerando reações correspondentes por parte do controlador. Os eventos externos são associados a contatos, e interpretados internamente através de bobinas. Cada contato associado a um evento externo é representado por um contato tipo P, que representa a subida da variável binária associada. Assim, a presença do contato tipo P evita que ações associadas a eventos com duração maior que um ciclo de varredura do programa sejam

executadas mais de uma vez. Analogamente, um evento externo também pode ser interpretado por um contato do tipo N, que representam a descida da variável associada. A figura 2.25 exemplifica esse procedimento aplicado à rede de Petri da figura 2.11.

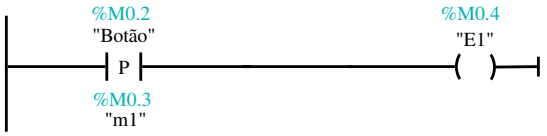


Figura 2.25: Módulo de eventos de um sinal de trânsito com botão de pedestre.

2.5.3 Módulo de Condições para o Disparo

Este módulo define as condições de habilitação para que as transições possam ser disparadas. Há diversas condições: um evento, um atraso, uma condição externa propriamente dita, como ocorre no sistema de sinal de trânsito. Na figura 2.26 está representado o módulo de condições para o disparo correspondente à rede de Petri da figura 2.11. Note que a transição t_4 só dispara quando o botão é pressionado na parte da noite, condição que pode ser interpretada como um sensor que "percebe" a falta de luminosidade. É representado pela variável "noite".

2.5.4 Módulo da Dinâmica

Este módulo atualiza o estado da rede de Petri, distribuindo as fichas nos lugares de acordo com o disparo de uma determinada transição. O módulo da dinâmica do sinal de trânsito é mostrado na figura 2.27. Note que foi colocado um contato normalmente fechado "t_4", na linha do disparo de t_1, de modo que t_1 só pode disparar quando o valor de t_4 for "falso", para resolver o conflito entre t_1 e t_4 com prioridade para a transição t_4.

2.5.5 Módulo de Ações

O módulo das ações associa uma bobina de saída ao lugar que possui uma ação instantânea. Desta forma, mediante o disparo de uma determinada transição, as ações associadas aos lugares de saída são executadas por meio de associações com bobinas. O tamanho do módulo das ações depende do número de lugares que possuem ações associadas. O

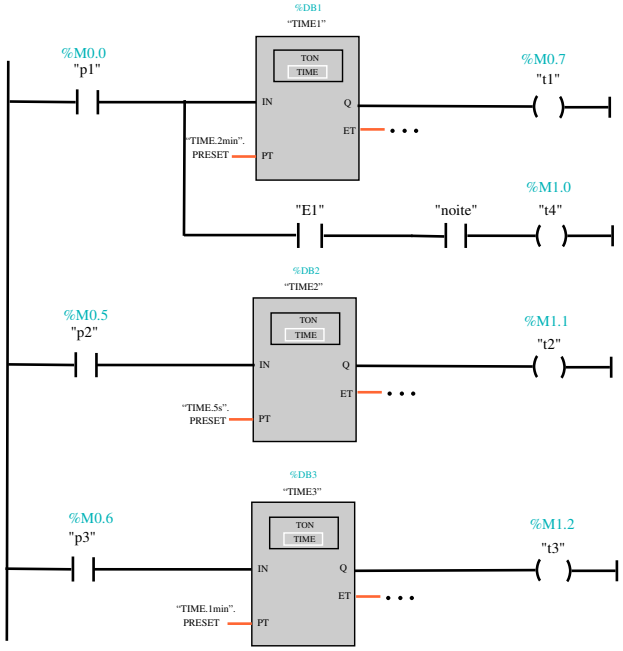


Figura 2.26: Módulo de condições para o disparo de um sinal de trânsito com botão de pedestre.

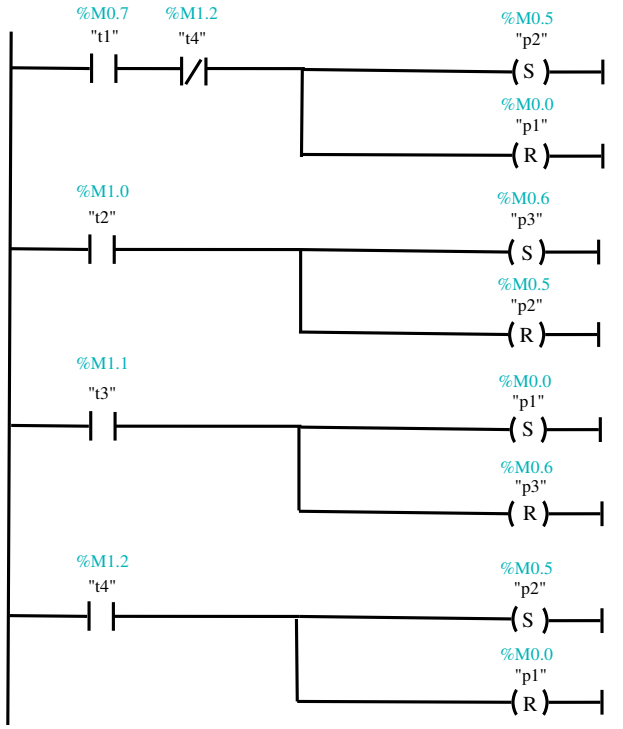


Figura 2.27: Módulo da dinâmica de um sinal de trânsito com botão de pedestre.

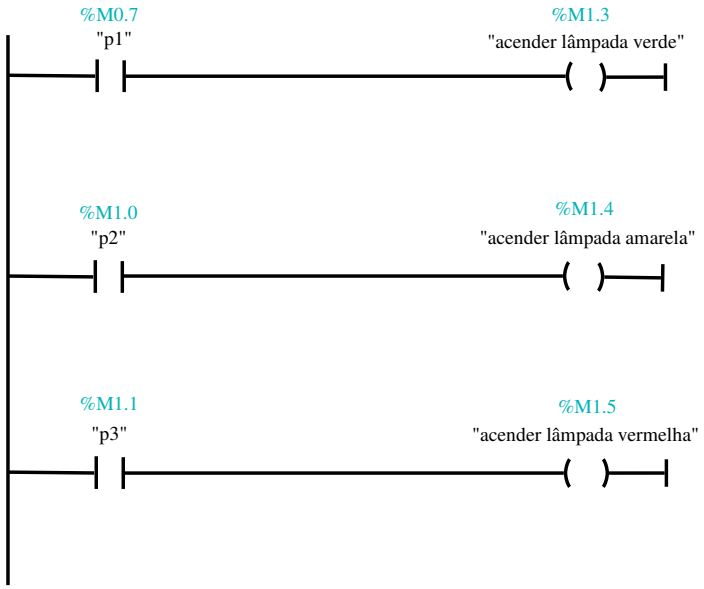


Figura 2.28: Módulo de ações de um sinal de trânsito com botão de pedestre.

módulo de ações do sistema semafórico é mostrado na figura 2.28. É importante observar que além de bobinas simples, as bobinas do módulo de ação podem ser tipo “SET” ou “RESET” também.

2.6 Controle Supervisório Modular

Considere, agora, o problema de se alterar o comportamento de uma planta de modo a satisfazer determinadas especificações. Nesse caso utiliza-se um supervisor que atuará sobre a planta para modificar o seu comportamento. A figura 2.29 mostra o diagrama esquemático de um sistema composto de uma planta G e de um supervisor S. O supervisor S atuará sobre a planta G, inibindo a ocorrência de eventos em G para adequar o sistema às necessidades do projetista. A esse tipo de controle dinâmico realimentado dá-se o nome de Controle Supervisório.

Em muitos casos, os sistemas são bastante complexos e há um alto custo computacional na realização de um único supervisor, isto é, uma única RPIC que controle o sistema por inteiro. Uma forma de contornar esse problema é considerar a modelagem de sistemas complexos a partir da união de modelos menores, isto é, dividir o sistema em vários módulos de modo a controlá-los simultaneamente. A esse tipo de controle dá-se o

nome de Controle Supervisório Modular [5], [6], [8]. No contexto desse trabalho, a ideia é construir as RPICs para cada módulo e os eventos em comum se encarregarão de unir os sistemas de forma a operar em conjunto. A figura 2.30 ilustra o conceito de controle supervisório modular, tal que ∪ denota a união dos conjuntos de eventos ativos.

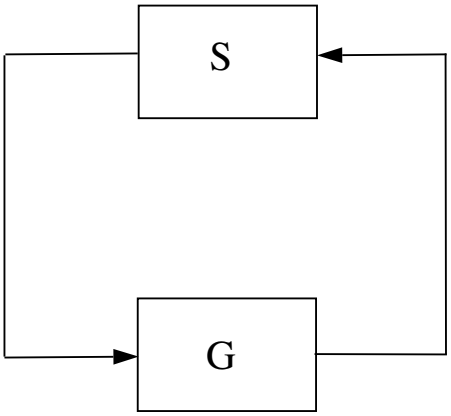


Figura 2.29: Exemplo de um diagrama de controle supervisório.

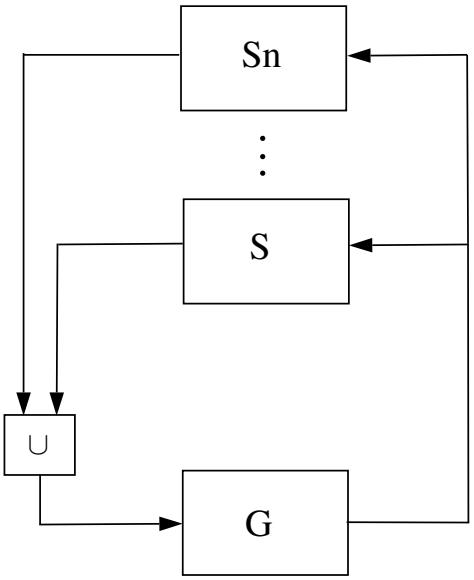


Figura 2.30: Exemplo de um diagrama de controle supervisório modular.

Capítulo 3

O Sistema de Manufatura

Este capítulo tem como objetivo principal descrever o funcionamento da planta e desenvolver o seu sistema de controle supervisório baseado nas especificações. A estrutura do capítulo é a seguinte. Na seção 3.1, há uma introdução ao projeto da célula de manufatura. Na seção 3.2, são apresentadas as especificações de projeto e na seção 3.3, tem-se a realização dos supervisores de cada componente, isto é, a obtenção de RPICs que modelam o comportamento controlado baseado nas especificações.

3.1 Projeto da Célula de Manufatura

Como este trabalho tem objetivo exclusivamente didático, a única preocupação é que a célula apresente complexidade suficiente para testar os conceitos apresentados no capítulo 2 e ratificar o método Moreira et al. [9]. A figura 3.1 mostra o diagrama da célula de manufatura considerada nesse trabalho e que foi montada de acordo com os equipamentos disponíveis no Laboratório de Controle e Automação (LCA).

O funcionamento básico da célula de manufatura acontece da seguinte maneira: a Esteira de Fornecimento (Ef) transporta peças até a Esteira Principal (Ep), que possui dois sensores. S_Ep, localizado no início de Ep que detecta a chegada de peças em Ep, e S_Ind, localizado no meio da esteira, e que serve para identificar o tipo da peça dentre os dois possíveis. As peças do tipo 1 devem ser levadas para $M1$ e as do tipo 2 devem ser levadas para $M2$. Após as peças chegarem nas máquinas, essas devem iniciar o processamento das peças. Após o processamento da peça por $M1$, ela deve ser transportada para $M2$ e após o processamento da peça por $M2$, ela deve ser levada de

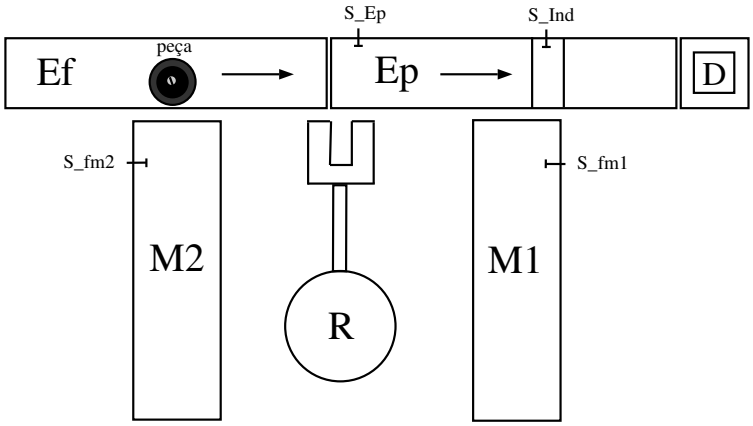


Figura 3.1: Diagrama da célula de manufatura.

volta para Ep, num local um pouco depois de onde ela foi retirada inicialmente. As duas máquinas $M1$ e $M2$ são esteiras que simulam um processamento ao levar a peça para frente e para trás num determinado tempo. Os sensores S_fm1 e S_fm2 detectam a presença de peças em $M1$ e $M2$, respectivamente, ao final de cada processamento. Finalmente, sempre que uma peça for detectada pelo sensor S_Ep a esteira de fornecimento Ef deve parar e sempre que uma peça for detectada pelo sensor S_Ind, a esteira principal Ep é quem deve parar.

No centro do sistema está o robô manipulador (R) que levará as peças para $M1$, $M2$ e de volta para Ep. O robô possui uma programação interna, que consiste em definir rotinas para que ele possa executar determinadas tarefas. Neste trabalho, são dadas quatro tarefas (rotinas) para o robô executar:

- Operação 1 (op1): pegar a peça do tipo 1 e transportar para a máquina 1 ($M1$);
- Operação 2 (op2): pegar a peça do tipo 2 e transportar para a máquina 2 ($M2$);
- Operação 3 (op3): transportar a peça de $M1$ para $M2$;
- Operação 4 (op4): transportar a peça de $M2$ para Ep.

O depósito D, localizado após a esteira Ep, recebe as peças já processadas.

3.2 Especificações

De acordo com a descrição do comportamento desejado para o sistema apresentada na seção anterior, tem-se que o sistema deve satisfazer às seguintes especificações:

1. A esteira de fornecimento Ef deve desligar quando o sensor S_Ep detectar a presença de uma peça, e religar quando esta peça for retirada da esteira principal pelo robô. Esta especificação garante que não haverá enfileiramento de peças em Ep, prejudicando a ação do robô.

2. A esteira Ep deve desligar quando o sensor indutivo, S_Ind, perceber a presença da peça e ligar quando o robô retirar a peça de Ep.

3. As peças do tipo 1 serão levadas para $M1$ e as do tipo 2 serão levadas para $M2$.

4. Após as peças chegarem nas máquinas, essas ligam.

5. Após o processamento da peça em $M1$, ela deverá ser transportada para $M2$.

6. Após o processamento da peça por $M2$, ela deverá ser levada de volta para a Ep, num ponto um pouco à frente de onde foi recolhida anteriormente.

7. As máquinas têm de estar vazias para receberem novas peças.

8. As operações do robô de retirada de peças das máquinas, $op3$ e $op4$, devem ter preferência sobre as operações de colocar peças. A única exceção é a situação em que há uma peça em $M2$, uma peça do tipo 2 esperando em Ep e uma peça do tipo 1 esperando em $M1$ após ter sido processada. Nesse caso a que está em $M2$ deverá ser transportada para Ep ($op4$), e em seguida, deve-se levar a peça do tipo 2 da esteira principal Ep para $M2$ ($op2$) em detrimento do transporte uma peça do tipo 1 que esteja esperando em $M1$.

3.3 Comportamento Controlado

Vamos considerar agora o desenvolvimento de RPICs para os diversos elementos da célula de manufatura de acordo com as especificações apresentadas na seção anterior.

3.3.1 Esteira de Fornecimento (Ef)

A dinâmica da RPIC da esteira de fornecimento, mostrada na figura 3.2, inicia-se no lugar $p1\_ef$ executando a ação "liga Ef". A transição $t1\_ef$ só dispara quando o sensor S_Ep, percebendo a presença de uma peça, envia ao CLP o sinal sep. Sendo assim, a ficha se desloca para o lugar $p2\_ef$, que desliga a esteira. A esteira só ligará novamente quando o robô pegar a peça em frente ao sensor indutivo, denominado evento $SI\downarrow$, disparando a transição $t2\_ef$. O evento $SI\downarrow$ é caracterizado pela descida da variável associada SI de 1 para 0. A descida de uma variável sempre será denotada neste trabalho com uma seta pra baixo, analogamente a subida por uma seta pra cima.

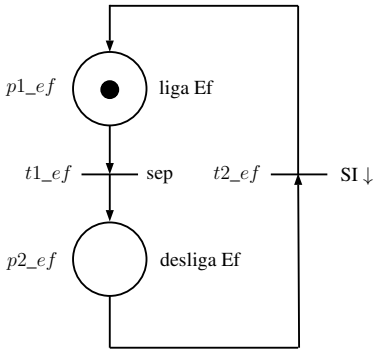


Figura 3.2: RPIC da esteira de fornecimento.

3.3.2 Esteira Principal (Ep)

A dinâmica da RPIC da esteira principal, mostrada na figura 3.3, tem início no lugar $p1\_ep$ executando a ação "liga Ep". A transição $t1\_ep$ só dispara quando o sensor S_Ind percebe a presença de uma peça no sensor indutivo e envia ao CLP o sinal $SI\uparrow$, que é caracterizado pela subida da variável associada ao evento SI. Sendo assim, a ficha se desloca para o lugar $p2\_ep$, que desliga a esteira. A esteira só ligará novamente quando o robô pegar a peça em frente ao sensor indutivo, disparando a transição $t2\_ep$. Isto é, na descida da variável SI de 1 para 0.

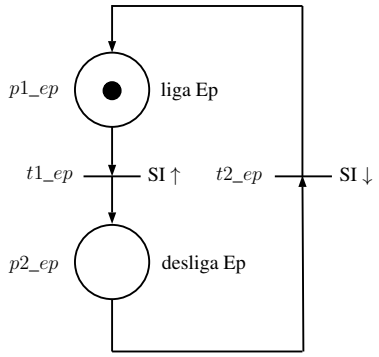


Figura 3.3: RPIC da esteira principal.

3.3.3 Máquina 1 (M1)

A dinâmica RPIC da máquina 1, mostrada na figura 3.4, tem início no lugar $p1\_m1$. Este lugar não possui ação associada a ele, representando apenas estado "máquina 1 vazia". A transição $t1\_m1$ dispara ao término da operação 1, $op1$, quando robô envia um sinal de fim de operação, $fop1$. Depois disso, a ficha se encontra no lugar $p2\_m1$, ligando $M1$. A máquina, que é uma esteira, permanecerá girando em um sentido até que tenha decorrido o $tempo\_M1$, que é de 7s. A seguir, a transição $t2\_m1$ dispara invertendo o sentido da rotação da esteira, em $p3\_m1$. A máquina gira em sentido contrário até que a peça se posicione na frente do sensor S_fm1, que enviará um sinal de fim de processamento $fm1$, e fará com que a máquina seja desligada em $p4\_m1$. A transição $t4\_m1$ não está associada a eventos, portanto sempre ocorre. Com a ficha estando em $p4\_m1$, a transição $t4\_m1$ fica habilitada e dispara, levando a ficha novamente para o lugar inicial $p1\_m1$.

3.3.4 Máquina 2 (M2)

A figura 3.5 mostra a RPIC da máquina $M2$ que, conforme pode-se perceber, é bastante semelhante à RPIC da máquina $M1$. Inicialmente a máquina encontra-se vazia e desligada no lugar $p1\_m2$, aguardando uma peça. Entretanto, diferentemente da RPIC da máquina $M1$, há um conflito entre as transições $t1\_m2$ e $t2\_m2$. Apesar da possibilidade da ocorrência de um conflito efetivo, não há possibilidade de um conflito real, uma vez que a máquina $M2$ pode receber peças de duas maneiras diferentes, que nunca acontecem simultaneamente. Assim, ou o robô executa a operação 3, levando peça tipo 1 da máquina

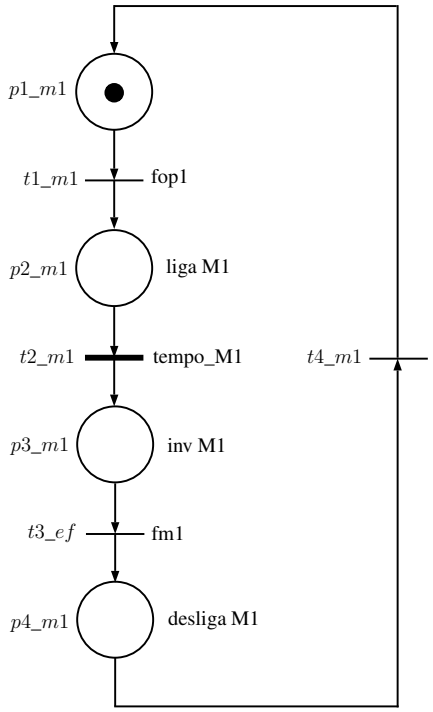


Figura 3.4: RPIC da máquina 1.

$M1$ para $M2$, ou executa a operação 2, transportando peça do tipo 2 de Ep para $M2$. Ao final de cada uma das operações, o robô envia um sinal de fim de operação, fazendo com que a máquina M2 ligue. Depois de ligada, a esteira inverterá o sentido após o $tempo\_M2$, que é de 4s e continuará funcionando até que a peça passe na frente do sensor S_fm2, que envia o sinal de fim de processamento $fm2$. Portanto, a máquina é desligada e depois retorna ao lugar $p1\_m2$ pelo disparo de $t5\_m2$.

3.3.5 Robô (R)

Dentre os componentes do sistema de manufatura, o robô é aquele que possui a RPIC mais complexa, pois é ele quem toma as decisões de como o sistema evoluirá como um todo. Sendo assim, suas ações têm consequências fundamentais em todos os outros componentes.

Conforme mostrado na figura 3.6, a RPIC do robô possui dois tipos de lugares: os que representam estados e os que indicam ações. Os lugares que representam estados são nomeados por três variáveis. A primeira indica a quantidade de peças em $M2$, 0 ou 1. A segunda indica a quantidade de peças em $M1$, porém com o seguinte detalhe: 0 representa $M1$ vazia, 1 indica $M1$ com máquina processando peça e 1* indica $M1$ com

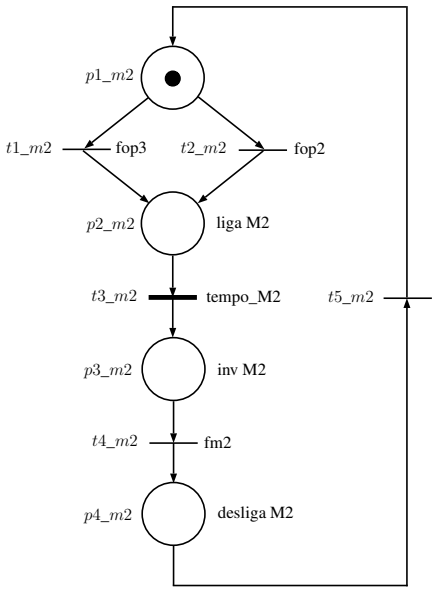


Figura 3.5: RPIC da máquina 2.

peça já processada e aguardando para ser retirada. A terceira variável possui três possíveis valores: 0 significa esteira principal sem peça, 1 com peça do tipo 1 e 2 com peça do tipo 2.

Os lugares que indicam ações são nomeados p_jr, com $j = 1, 2, ...17$. Eles estão associados às quatro operações possíveis que o robô pode executar como dito na subseção 3.1. As operações são denotadas por *opi*, $i = 1, 2, ...4$. Os 17 lugares de ação unidos aos 15 de estado somam 32 lugares e as transições nomeadas t_ir, $i = 1, 2, ...45$, são 45 no total. A maioria dessas transições tem seu disparo associado a eventos. Os eventos $c1$ e $c2$ indicam a chegada de peça do tipo 1 e 2 , respectivamente. Além disso, $fm1$ e $fm2$ representam o envio dos sinais de fim de processamento de cada máquina, conforme já mencionado nas subseções 3.3.3 e 3.3.4. Por fim, há os eventos de final de operação $fopi$, $i = 1, 2, ...4$.

Observação 3.1 *Os lugares que indicam ações na figura 3.6 são ao todo 17, porém na verdade são 34, isto é, o dobro. Isto ocorre porque cada lugar tem como ação uma operação do robô. Como o tempo máximo de varredura das entradas do robô é de 3s, o comando para ativar a operação deve permanecer também ligado por 3s e depois pode ser desligado. Conforme mostrado na figura 3.7, esse problema foi superado implementando-se dois lu-*

gares para cada operação ao invés de um, uma para ligar e outro para desligar o sinal de comando. Portanto, adicionando os lugares e transições extras, a RPIC do robô passa a ter, ao todo, 49 lugares e 62 transições.

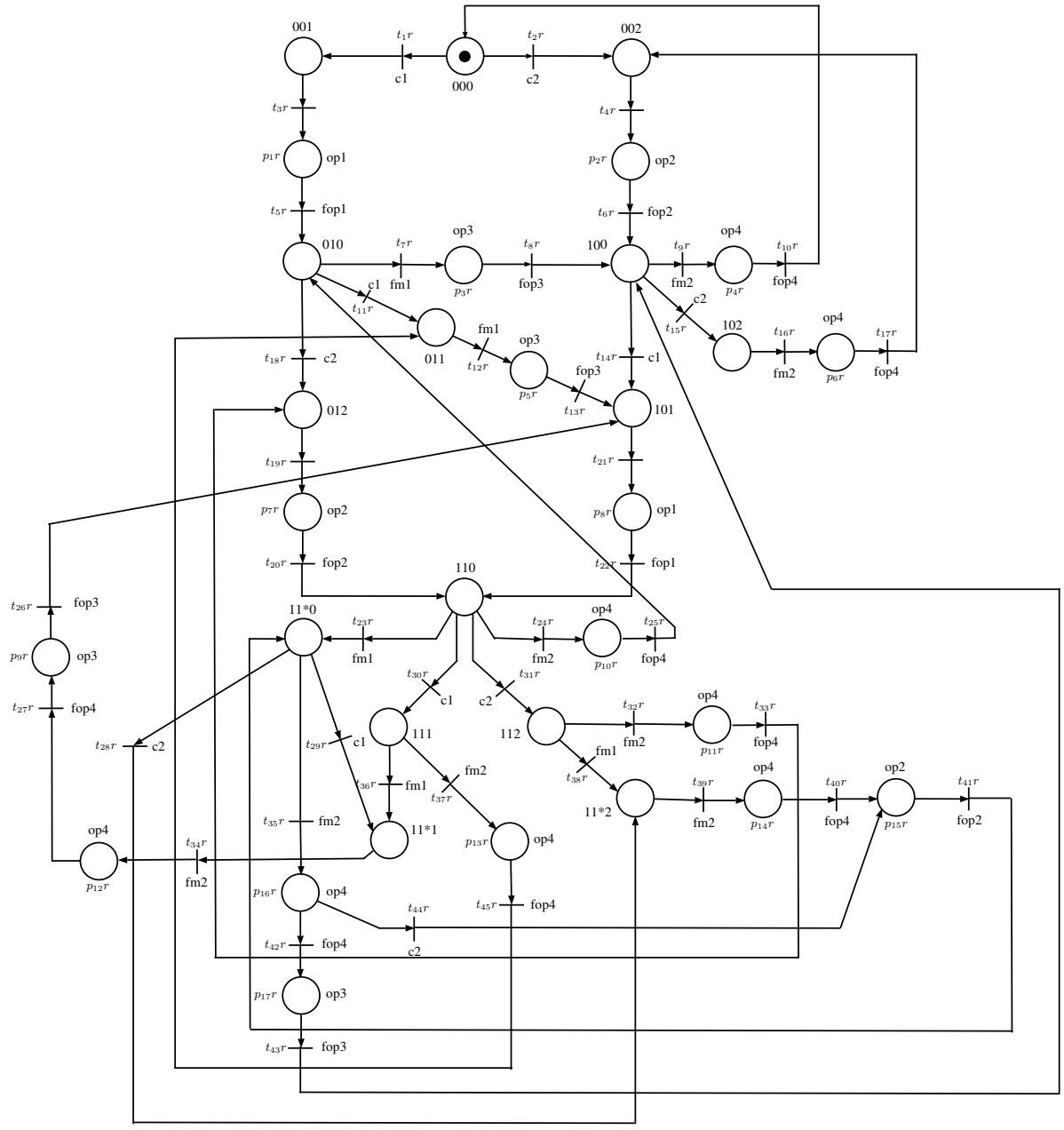


Figura 3.6: RPIC do robô.

A RPIC da figura 3.6 contém sete conflitos estruturais, listados na tabela 3.1, sendo que em 6 há a possibilidade de um conflito real. Eles podem ocorrer nos lugares: 010,

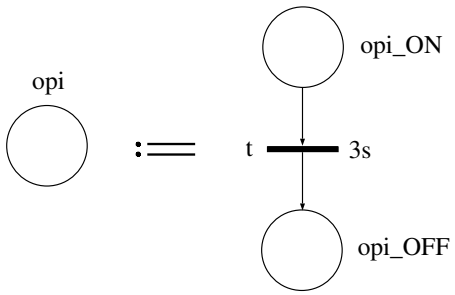


Figura 3.7: Lugares para cada operação.

Tabela 3.1: Conflitos e prioridades da RPIC do robô.

Conflito estrutural	Prioridade
$\{000, t_1r, t_2r\}$	-
$\{010, t_7r, t_{11}r, t_{18}r\}$	$\{(t_7r, t_{11}r \vee t_{18}r)\}$
$\{100, t_9r, t_{14}r, t_{15}r\}$	$\{(t_9r, t_{14}r \vee t_{15}r)\}$
$\{110, t_{23}r, t_{24}r, t_{30}r, t_{31}r\}$	$\{((t_{23}r, t_{24}r), t_{30}r \vee t_{31}r)\}$
$\{11^*0, t_{28}r, t_{29}r, t_{35}r\}$	$\{(t_{35}r, t_{28}r \vee t_{29}r)\}$
$\{111, t_{36}r, t_{37}r\}$	$\{(t_{36}r, t_{37}r)\}$
$\{112, t_{32}r, t_{38}r\}$	$\{(t_{38}r, t_{32}r)\}$

100, 110, 11*0, 111 e 112. Os eventos de chegada de peça, c1 e c2, nunca podem ocorrer ao mesmo tempo, portanto não há um conflito real em 000. Uma peça pode chegar a Ep ao mesmo tempo em que ocorra um fim de processamento em uma das máquinas. Em relação a esse conflito real, prioriza-se os eventos de fim de processamento. Um caso pouco provável, mas possível de ocorrer, até pelos tempos distintos de processamento utilizados neste trabalho, é a ocorrência simultânea dos eventos fm1 e fm2 nos lugares 111 e 112. Nesse caso foi priorizado o fim de processamento da máquina 1. A implementação das prioridades que solucionam os conflitos no diagrama ladder serão discutidas na seção 4.2 cujas prioridades são também mostrados na tabela 3.1. O sinal $\vee$ significa "ou", isto é, não há prioridade entre as transições, uma vez que não há possibilidade delas ocorrerem ao mesmo tempo.

Capítulo 4

Implementação Prática do Sistema de Manufatura

Este capítulo tem como objetivo descrever os aspectos físicos dos componentes da célula de manufatura, programação de software, bem como sua montagem. Primeiramente, na seção 4.1, descrevem-se os materiais utilizados no trabalho, a programação do robô e as interfaces entre os subsistemas. Em seguida, na seção 4.2, são discutidos detalhes dos diagramas Ladder mostrados no apêndice C. Experimentos realizados com o sistema montado no LCA são mostrados na seção 4.3.

4.1 Descrição dos Equipamentos do Projeto

4.1.1 Robô

O robô utilizado neste trabalho é um braço robótico AL5D, da LynxMotion, ilustrado na figura 4.1. Além da abertura e fechamento da garra, ele possui quatro graus de liberdade: a base, o ombro, o cotovelo e o movimento vertical do pulso. Cada eixo tem movimento de 180^o, e é controlado por um servomotor. Sem a bateria, o robô pesa cerca de 960g, e pode carregar aproximadamente 385g. A abertura da garra é de aproximadamente 3,2cm. O controle é realizado através de uma placa SSC-32, mostrada na figura 4.2. Ela possui um microcontrolador Atmel ATMEGA168-20PU, 32 saídas e 4 entradas que podem ser usadas como entradas analógicas ou digitais. Os servos são alimentados com 6V, ao passo que a placa de controle é alimentada com uma bateria de 9V.

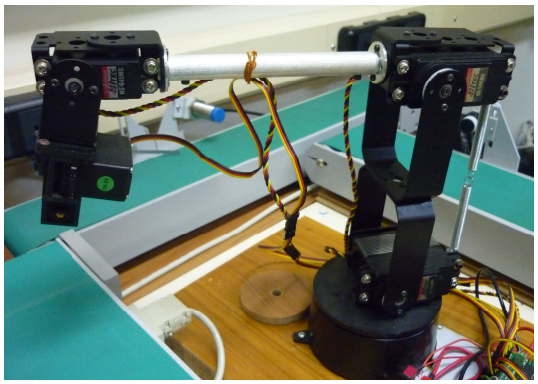


Figura 4.1: Robô utilizado no projeto.

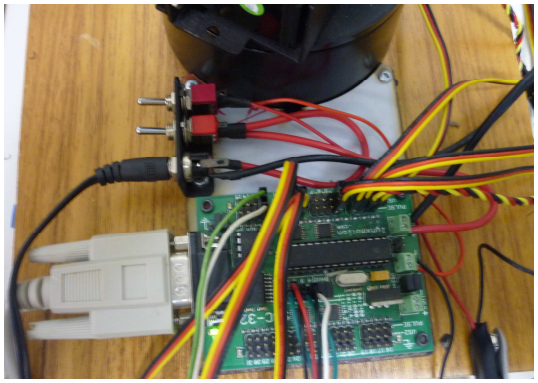


Figura 4.2: Placa SSC-32.

A programação do robô é feita no software *Lynxmotion RIOS SSC-32 V1.06*, disponibilizado pelo fabricante. Nele podemos definir passos, sequências e projeto. O programa possui um eixo de coordenadas xyz que tem como origem o ponto central da base do robô. Utilizando a tela mostrada na figura 4.3, é possível definir uma posição no espaço usando os botões no canto superior direito da tela para movimentar a base, o ombro, o cotovelo, pulso ou fechamento e abertura da garra. Antes ou após colocar o robô numa certa posição, pode-se ajustar a velocidade com que o robô chega a esta posição. A velocidade pode variar numa escala de 1 a 100.

A interpolação de vários passos define uma sequência, isto é, uma operação. É importante destacar que quanto mais passos forem definidos em uma sequência, mais precisa será sua trajetória. Por exemplo, ao gravar uma sequência para levar a garra de um ponto inicial padrão até a frente de um sensor com apenas um ou dois passos, ele irá percorrer um menor caminho possível, contudo é possível que esbarre no sensor,

ocasionando problemas. Por isso a estratégia neste trabalho foi definir quatro sequências diferentes contendo de 10 a 12 passos, com deslocamentos velozes até chegar em uma peça e vagarosos nos passos necessários para agarrar as peças, que são visivelmente os mais críticos. No sistema desenvolvido neste trabalho, as operações são muito semelhantes, diferindo apenas no lugar onde começam e terminam.

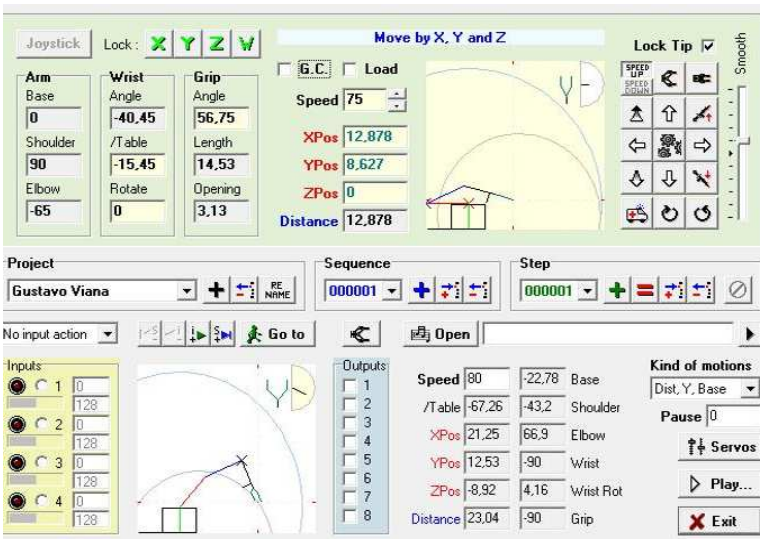


Figura 4.3: Tela de programação do robô.

As operações começam com o robô posicionado à frente do local onde ele pegará a peça. Em seguida, ele desce, ajeita a garra, e pega a peça. Após pegar a peça, o robô a desloca horizontalmente para que ele não acerte o sensor ao subir. Por fim, o braço ergue a peça, gira para o seu destino, e desce para soltá-la. No penúltimo passo, quando a garra solta a peça, o robô envia o sinal de fim de operação. Isto é feito marcando com um visto a "caixa" de alguma saída, "OUTPUT" na figura 4.3. O sinal não é enviado no último passo porque, neste caso, o robô continuaria enviando o sinal até que outra operação se iniciasse. Então no último passo, o robô retorna a posição de espera. As rotinas de programação do robô se encontram no apêndice A.

Para executar as tarefas desejadas, o programa possui uma outra interface mostrada na figura 4.4. Para acessá-la, basta apertar o botão "PLAY" na tela da figura 4.3. Na interface da figura 4.4, pode-se programar um projeto utilizando comandos de lógica e as sequências. Os comandos de lógica que o software disponibiliza são: "Do...While",

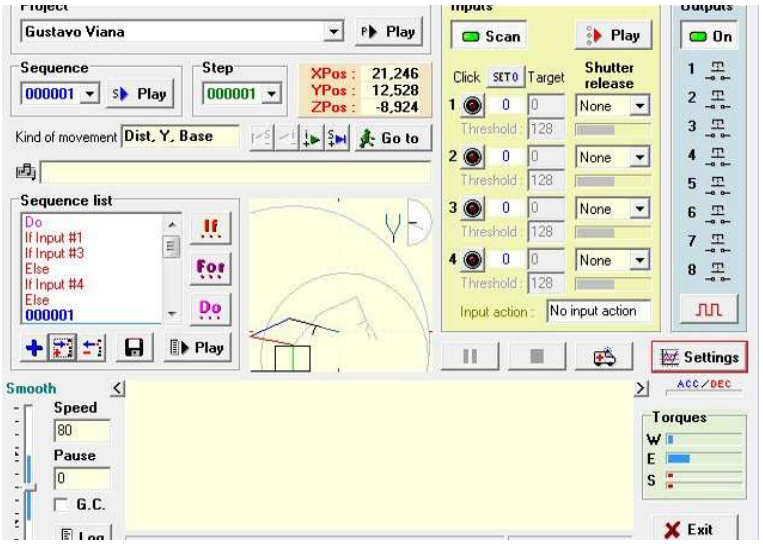


Figura 4.4: Tela de execução de um projeto do robô.

"If..else...End", "Next", "For" e "Counter". Para executar as 4 operações necessárias foi usada a lógica apresentada no apêndice B.

O robô possui 4 entradas, que serão ligadas ao CLP, conforme será detalhado na seção 4.1.5. A entrada 2, nomeada INPUT #2, está sempre ligada e isso é aproveitado da seguinte forma: um laço "Do...While INPUT #2" fará com que o programa fique sempre executando o que ocorre dentro do laço. Considerando 0 como "entrada desligada" e 1 como "entrada ligada", os valores lógicos das outras três entradas podem gerar 8 combinações distintas. Serão usadas apenas quatro delas. A operação 1 só ocorre se Input #1 estiver ligada, e Input #3 e #4 desligadas. A operação 2 só ocorre se Input #3 estiver ligada, e Input #1 e #4 desligadas. A operação 3 acontece se Input #4 estiver ligada, e Input #1 e #3 desligadas. Por fim, a operação 4 só ocorre se Input #1 e #3 estiverem ligadas, e Input #4 desligada.

A programação do robô feita desta maneira foi um avanço em relação ao projeto anterior feito em autômato [12], já que o robô ganhou velocidade para realizar as operações. Isto porque anteriormente somente uma entrada foi usada, criando a necessidade do uso de um contador para dar tempo ao CLP de enviar a quantidade de pulsos necessária para diferenciar cada operação. Isto fez com que o robô respondesse mais lentamente a uma ordem de execução de tarefa.

4.1.2 Esteiras

As esteiras utilizadas neste trabalho são correias transportadores do fabricante *Interditatic*. Elas possuem dois motores CC com alimentação de 24V. Possuem cerca de 60cm e em regime permanente consomem uma corrente de 120mA. A movimentação das esteiras pode ser feita para ambos os sentidos através de uma entrada que inverte o sentido. Porém, sua velocidade não pode ser manipulada. A figura 4.5 mostra a esteira que representa a máquina $M2$.

A base das esteiras, mostrada na figura 4.6, possui uma estrutura que permite a fixação dos sensores ao longo de sua extensão, bem como outros dispositivos que possam ser adaptados ao local. Pode-se também colocar as esteiras em sequência, de modo que uma peça passe de uma para a outra, como foram colocadas a esteira de fornecimento e principal, conforme mostrado no esquema da figura 3.1. Contudo, este tipo de arranjo faz com que haja uma folga entre elas, criando a possibilidade de uma peça ficar presa nesse vão. Esse problema foi superado pelo uso de um pedaço de madeira em formato de cruz, conforme mostra a figura 4.7, que foi encaixado entre as esteiras de forma a cobrir o buraco e permitir a passagem das peças. Além disso, exclusivamente na esteira principal foi encaixada uma guia, mostrada na figura 4.8, para conduzir as peças de forma a passar na frente do sensor indutivo.

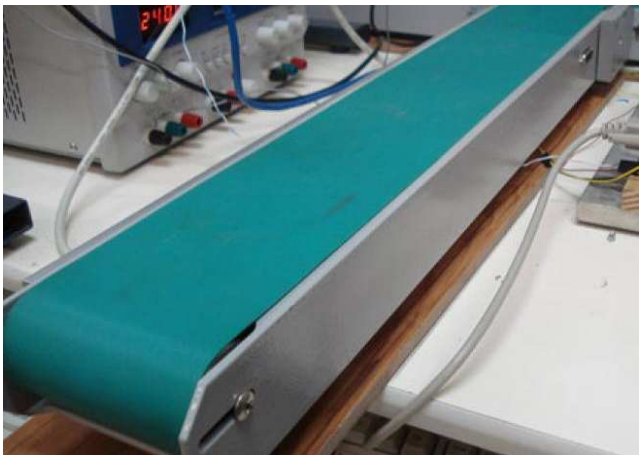


Figura 4.5: Esteira utilizada no projeto.

Figura 4.6: Base para encaixe de equipamentos auxiliares.

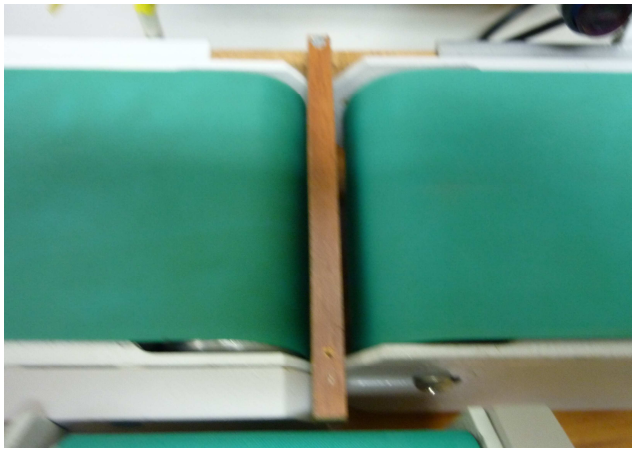


Figura 4.7: Madeira para encaixe das esteiras principal e de fornecimento.

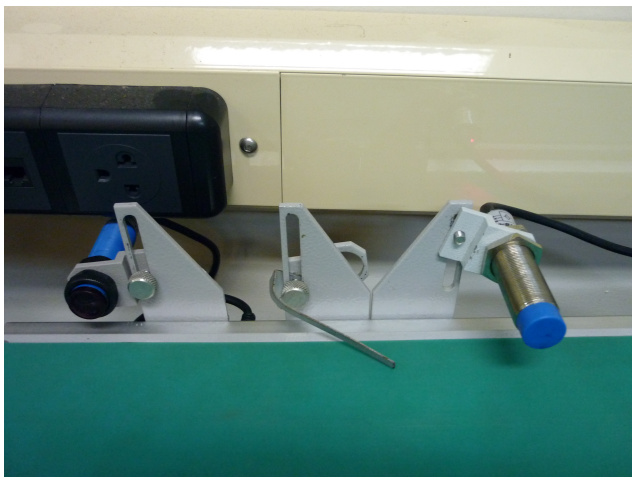


Figura 4.8: Guia para conduzir as peças até o sensor indutivo.

4.1.3 Sensores e Peças

O projeto conta com 3 sensores fotoelétricos da *Interdidatic*, 2 da cor cinza e 1 da cor azul, que detectam a presença da peça. Eles possuem um alcance de 10 a 30cm e, ao detectarem um objeto, enviam um sinal de 24V. Os sensores possuem uma base que permite encaixá-los às esteiras, e um cabo para conectar às mesmas.

Além destes sensores, foi utilizado também um sensor indutivo analógico, modelo I18-ANV, da Metaltex. Ele é usado para reconhecer a presença e identificar os diferentes tipos de peças. O seu sinal de saída é proporcional à distância do objeto detectado e funciona da seguinte maneira: para distâncias iguais ou maiores a 8mm, o sinal é de 10V; à medida que a distância diminui, o sinal também diminui, chegando a 0V quando a distância for nula. O cabo do sensor permite acesso a três fios, permitindo a aquisição do sinal e a alimentação, que deve estar na faixa de 10 a 36V. O sensor de presença é mostrado na figura 4.9, enquanto o indutivo é apresentado na figura 4.10.

As peças, ilustradas na figura 4.11, são feitas com cilindros de aço galvanizado, fixados com durepox a um suporte circular de madeira. No centro há um parafuso, que é onde a garra do robô se prende. A diferenciação do sensor entre os tipos de peça é feita através da altura delas, sendo que a peça 1 (cor preta) é ligeiramente menor do que a peça 2 (cor verde).

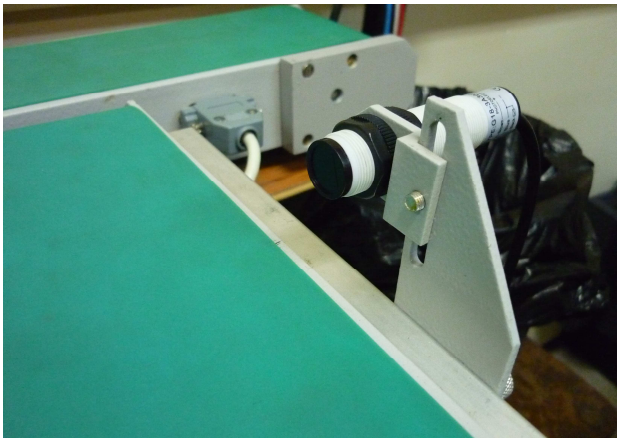


Figura 4.9: Sensor de presença utilizado no trabalho

Figura 4.10: Sensor indutivo *Metaltex*.

Figura 4.11: Peça tipo 2 (esquerda) e peça tipo 1 (direita).

4.1.4 Controlador Lógico Programável (CLP)

O controlador lógico utilizado no projeto foi a CPU modelo s7-1200 Siemens, mostrado na figura 4.12. Ele é caracterizado pelo seu conceito de instalação versátil e flexível, aliado a um elevado desempenho e a um design extremamente compacto. Sua programação, comunicação e comissionamento são especialmente simples e rápidos. Sua alimentação é em corrente alternada 120-240 VAC. Contém internamente um convesor CA-CC, possibilitando assim uma saída de 24V para alimentar os outros componentes: o *hub* e uma tela HMI (interface homem-máquina). O CLP possui 14 entradas digitais 24V e duas entradas analógicas. Contém um módulo externo com três saídas analógicas e 10 saídas digitais. Ele também possui entrada para módulos de saídas digitais, caso haja a necessidade de

se trabalhar com mais saídas. Para maiores informações sobre o CLP s7-1200 consultar a referência [14].

A aquisição desse CLP pelo Laboratório de Controle e Automação UFRJ também foi um grande avanço para esse trabalho. No trabalho anterior [12], foi usado o CLP da Rockwell (RSLogix 500), que possui uma quantidade de saídas muito reduzida. Isso ocasionava muitos problemas e adaptações, deixando o trabalho menos robusto.

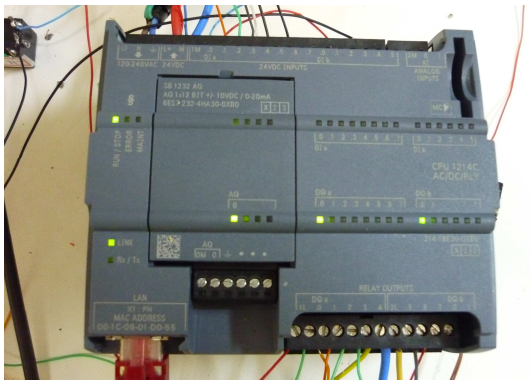


Figura 4.12: CLP Siemens s7-1200.

4.1.5 Montagem e Esquemas de Ligações

A disposição dos equipamentos que compõem a célula de manufatura é ilustrada na figura 4.13. As esteiras Ef e Ep ficam alinhadas, com o robô em frente delas. As esteiras que representam M1 e M2 ficam cada uma de um lado do robô, de modo a permitir que ele alcance todas as esteiras satisfatoriamente. As conexões dos equipamentos entre si e com o CLP foram feitas com cabos telefônicos e fios avulsos.

A alimentação do sensores e das esteiras é feita pelo *rack* mostrado na figura 4.14. Tendo em vista a disposição dos equipamentos, as esteiras de fornecimento de peças operam no sentido inverso, através de uma conexão no *rack*. O *rack* possui 4 módulos, sendo que em cada módulo há uma placa que converte a tensão da rede 127 VCA em 24VCC para alimentar cada esteira.

Interface entre o Robô e o CLP

O controlador precisa receber sinais e enviar comandos para o robô. Os comandos de operação são enviados através das saídas digitais Q0.5, Q0.6, Q0.7. A figura 4.15 apresenta

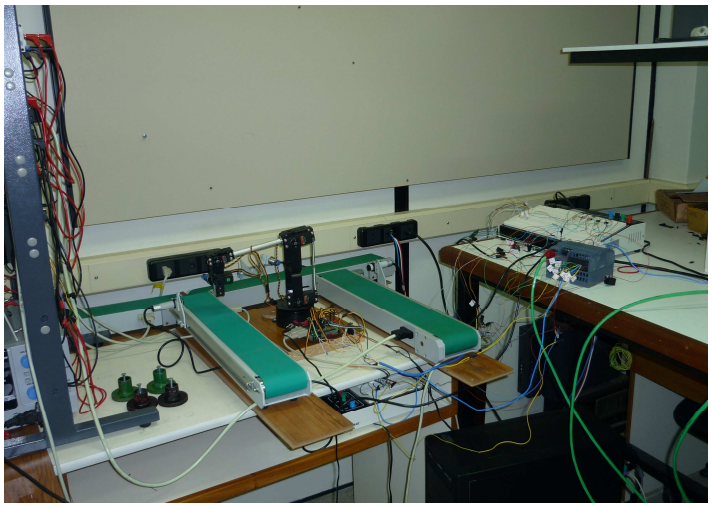


Figura 4.13: Bancada em que o sistema foi implementado.

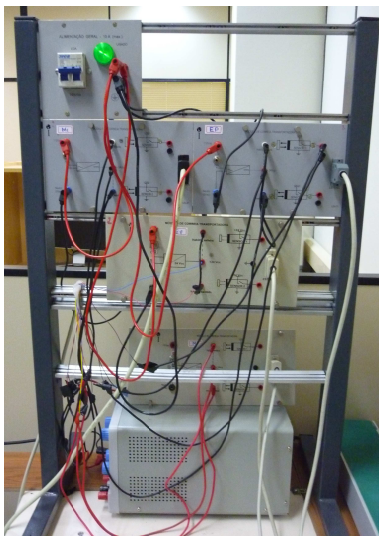


Figura 4.14: Rack usado para alimentar as esteiras.

o diagrama de ligações entre a placa do robô e o CLP. No diagrama as entradas da placa A, B, C e D são, respectivamente, Input #1, Input #2, Input #3 e Input #4. O sinal "+" corresponde a +5V e o sinal "-" é o terra (GND) da placa. Como a entrada B fica sempre ligada, ela não é usada. As entradas utilizadas são acionadas através de um pulso de corrente. Quando a chave, que é interna ao CLP, está aberta (saída do CLP desligada) a entrada está em curto com +5V, e portanto, não há pulso de corrente; consequentemente, a entrada não é excitada. Se a chave estiver fechada, a entrada estará em curto com o "GND". Sendo assim, haverá uma diferença de potencial de 5V no resistor de $10k\Omega$, gerando um pulso de corrente de 0,5mA que aciona a entrada do robô. Para execução

deste esquema de interface foi utilizado um protoboard PRONT-O-LABOR, modelo PL-556k.

Ao término de execução de cada operação, o robô envia um sinal de fim de processamento. Esses sinais são enviados para uma das saídas numeradas de 8 a 15 na placa. Essas saídas possuem correpondente direto no software, porém a numeração vai de 1 a 8. Portanto, ao programar a utilização da saída 1 no software será na verdade a 8 no hardware, 2 será 9 e assim por diante. As saídas físicas utilizadas foram: 8, 9, 10 e 12. Os sinais seguirão respectivamente para as entradas: I0.1, I0.2, I0.3 e I0.4, que correspondem ao fim das operações 1, 2, 3 e 4.

Um detalhe importante é que o sinal de saída do robô é fixo em 5V e o CLP só excita suas entradas com sinais superiores a 13V. Portanto fez-se necessário o uso do circuito amplificador não inversor, ilustrado na figura 4.16 cuja montagem foi feita utilizando uma protoboard PRONT-O-LABOR, modelo PL-556k. Nessa protoboard são recebidos os quatro sinais do robô e amplificados em direção ao CLP. Esse protoboard também foi aproveitado para distribuir as saídas de 24VCC do CLP para alimentar a interface entre esteiras e CLP, que será descrita na seção seguinte. Os 4 circuitos de amplificação são mostrados na figura 4.17. Cada amplificador possui um CI 741 que é alimentado por +15V e -15V. O resistor de entrada R1 é de 510Ω e o de saída R2 é de $1k\Omega$. Portanto com uma tensão de entrada de 5V obtém-se uma tensão de saída de 14.8V, o suficiente para acionar a entrada do CLP.

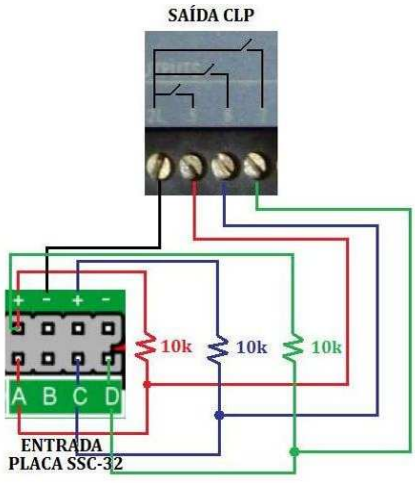
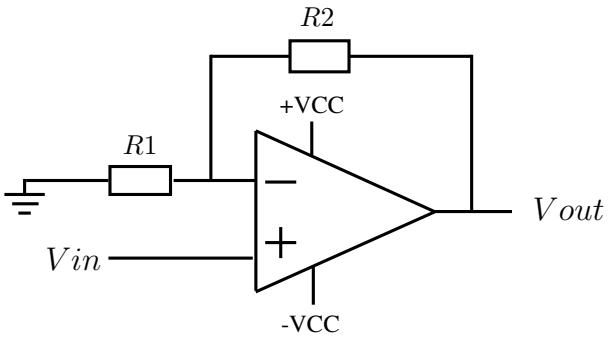


Figura 4.15: Diagrama de ligações entre placa e CLP.



$$\frac{Vout}{Vin} = \frac{R1+R2}{R1}$$

Figura 4.16: Esquema de um amplificador não inversor.

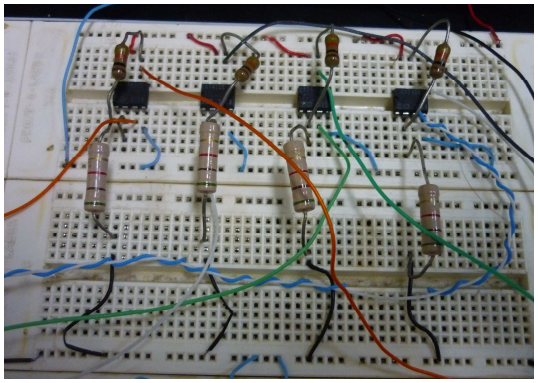
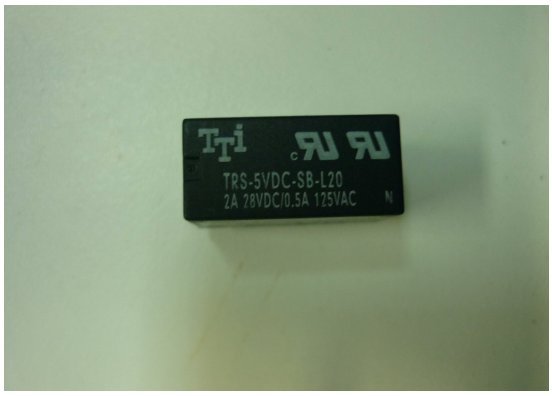


Figura 4.17: 4 circuitos de amplificação não inversora.

Interface Esteiras, Sensores e CLP

Conforme dito anteriormente, as esteiras são alimentadas através de um *rack*. Apesar do *rack* possuir uma saída de 24V, assim como o CLP, a interface entre o *rack* e o CLP é feita através de relés eletromecânicos que conferem isolamento entre os dois sistemas. Isto porque não é recomendável que se coloque fontes em paralelo de origens diferentes. Por mais que a tensão seja especificada como igual, na prática uma pequena diferença pode fazer circular uma corrente capaz de danificar algum equipamento. Foram utilizados os seguintes relés: o AX1RC3-24VDC, da *Metaltex*, e o TRS-5V, da *TTi*, conforme mostrado na figura 4.18. Estes relés possuem uma bobina que, ao ser alimentada com a tensão especificada, faz com que a chave, que curto circuitava o pino comum ao pino NF, conecte o pino comum ao pino NA. A figura 4.19 mostra a pinagem dos relés utilizados neste



(a)

(b)

Figura 4.18: Relé 5V (a) e relé 24V (b).

trabalho.

Para realizar a interface de $M1$ e $M2$, foi utilizado o relé de 24V; cujo esquema de montagem é mostrado na figura 4.20. A saída comum $1L$ recebe a alimentação de 24V do próprio CLP, enquanto o neutro M vai para a bobina do relé superior. Quando $Q0.1$ é ligada ocorre um curto com $1L$ e a bobina recebe 24V. Sendo assim, o pino comum 2 é curto-circutado com o pino 5 e a esteira é habilitada. A esteira ficará ligada se, e somente se, $Q0.1$ estiver ligada. Caso a saída $Q0.2$ seja acionada, a bobina do relé inferior é alimentada invertendo-se o sentido de rotação da esteira. A interface de Ep é similar à da figura 4.20. Porém como a esteira principal não precisa inverter o sentido de rotação, há apenas um relé para ligá-la. Todas essas esteiras demandam 5 saídas do CLP: Q0.1 e Q0.2 habilitam e invertem o sentido de M1, respectivamente, Q0.3 e Q0.4 habilitam e invertem o sentido de M2 e Q0.0 liga a esteira Ep.

Para ligar a esteira Ef, foi necessária a utilização do relé de 5V. Como as cinco saídas que têm 1L (alimentado por 24V) como ponto comum foram ocupadas pelas outras esteiras, a esteira de fornecimento foi ligada à saída Q1.0, que tem 2L (alimentado por 5V vindo da placa do robô) como ponto comum. O esquema de ligação da esteira de fornecimento é mostrado na figura 4.21.

As esteiras possuem uma entrada para sensores que é ligada ao *rack*. Como as saídas de sensor no *rack* apresentaram defeito, apenas o sensor S_Ep foi alimentado pelo *rack*. Os demais sensores foram alimentados pelo CLP.

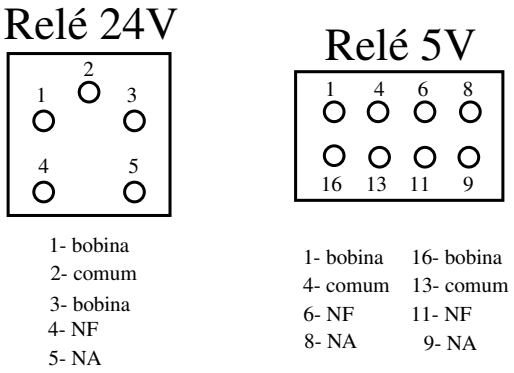


Figura 4.19: Pinagem dos relés utilizados no trabalho.

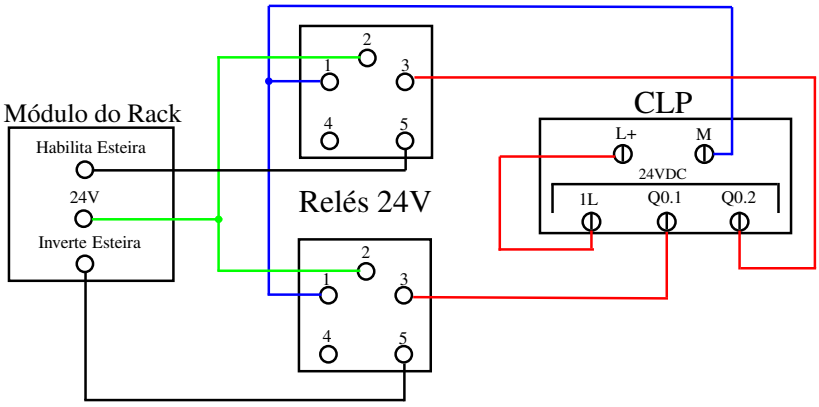


Figura 4.20: Esquema de ligações da interface entre *rack* e CLP com relé 24V.

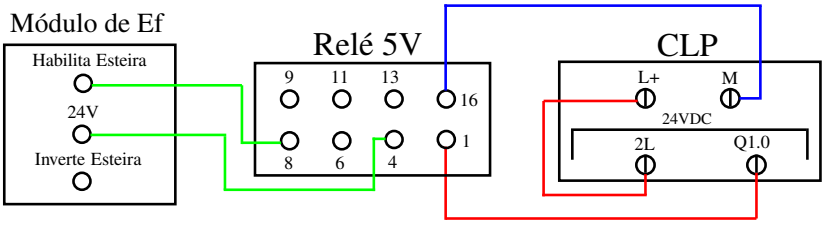


Figura 4.21: Esquema de ligação da interface entre módulo de Ef e CLP.

4.2 Descrição dos Diagramas em Ladder

Os diagramas Ladder utilizados para a programação do controle supervisório estão mostrados no apêndice C. Cada RPIC apresentada na seção 3.3 foi convertida em Ladder por meio do método explicado na seção 2.5. Os diagramas funcionam simultaneamente tal que os eventos em comum se encarregam de uni-los.

4.2.1 Esteira de Fornecimento

No diagrama Ladder da esteira de fornecimento Ef, mostrado no apêndice C.1, o módulo de inicialização "seta" o lugar $p1\_ef$, que equivale à marcação inicial da RPIC da figura 3.2. No módulo de eventos são recebidos os sinais externos à esteira e fundamentais para o funcionamento correto da mesma. A subida de 0 para 1 do sinal *sep*, que é enviado do sensor localizado na esteira principal, energiza a bobina associada à variável $Esep$.

Para uma leitura mais precisa do sensor indutivo foi colocado no diagrama Ladder um temporizador chamado "ajeitando peça". Ele causa um atraso de 500ms que é o tempo necessário para que após o sensor reconhecer que há uma peça, ele consiga também diferenciá-las. Quando chega uma peça no sensor indutivo, a bobina associada à variável SI é energizada. Quando o robô retira peça, a variável SI vai de 1 para 0 e energiza a bobina Er.

O módulo de condições de disparo tem transições associadas aos eventos $Esep$ e Er. Isto significa que as transições só poderão disparar se tais eventos ocorrerem. Os eventos são associados a um contato NA. O módulo da dinâmica possui transições associadas a contatos NA, de forma que ao dispararem retiram (*reset*) a ficha dos lugares de entrada e marcam (*set*) os lugares de saída. O módulo de ações tem como objetivo fazer a conexão entre as bobinas associadas aos lugares de ação e as saídas do CLP. Em relação ao diagrama Ladder de Ef, o lugar $p1\_ef$ liga a esteira, enquanto $p2\_ef$ desliga.

4.2.2 Esteira Principal

O diagrama Ladder da esteira principal Ep, mostrado no apêndice C.2 é análogo ao diagrama de Ef, porém o módulo de eventos, além de possuir uma bobina associada à descida da variável SI (Er) também possui uma bobina associada à subida de SI (Ec).

4.2.3 Máquinas M1 e M2

Os diagramas Ladder das máquinas $M1$ e $M2$ estão representados nos apêndices C.3 e C.4. Note que as máquina $M1$ e $M2$ tem como lugar inicial um lugar que não está associado a nenhuma ação. Esse lugar existe apenas para que a máquina, estando sem

peça, espere que o robô forneça a peça para começar o processamento. É importante ressaltar que os eventos fim de operação (fop) são definidos no diagrama do robô. A linha de comando que recebe o sinal externo de fim de operação não foi repetida no diagrama Ladder das máquinas pois, nesse caso, haveria uma redundância no diagrama. Outro ponto a ser comentado é que no módulo de ações das máquinas o primeiro lugar de ação tanto liga a máquina quanto inverte o sentido de movimento da esteira, uma vez que a peça inicialmente se desloca no sentido contrário ao sentido padrão. Após um tempo de processamento, o Ladder desfaz a inversão do sentido, "resetando" a saída associada a inversão de sentido.

4.2.4 Robô

O diagrama Ladder do robô, mostrado no apêndice C.5 é o mais complexo dentre todos os diagramas, devido à grande quantidade de lugares e transições. O módulo de inicialização tem como objetivo colocar a ficha no lugar de estado "000". O módulo de eventos possui todos os eventos de fim de operação tal que no penúltimo passo da operação, o robô envia o sinal que energiza a bobina $Efopi$, $i = 1, 2, ...4$.

Para facilitar a programação, o módulo de condições para o disparo do diagrama Ladder do robô está organizado da seguinte maneira: na primeira parte são implementadas as condições relativas a todas as transições que possuem os lugares de estado como entrada, e em seguida, são implementadas as condições relativas às transições que possuem lugares de ação como entrada. Baseado na ideia de que existe um lugar para ligar e outro para desligar o sinal de ordem para executar uma ação, o módulo de ação é entendido facilmente. Os lugares opi_ON "setam" as bobinas associadas às entradas do robô, enquanto os lugares opi_OFF as "resetam", respeitando a lógica de programação mostrada no apêndice B.

Outro detalhe importante do diagrama Ladder do robô é que os eventos de fim de processamento de peça são lidos como contatos NA e não de borda de subida (contato tipo P). Isto ocorre porque se o robô estiver executando uma operação e acabar o processamento em qualquer uma das máquinas ele não reconhecerá que ocorreu, pois sua RPIC estará num lugar de operação e não de estado. Colocando um contato NA, o sensor ainda

estará ligado e consequentemente o evento fim de processamento estará ocorrendo até que a ficha da RPIC se desloque para um lugar de estado.

Em relação aos conflitos reais listados na tabela 3.1, eles foram solucionados no diagrama Ladder do robô pela adicão de contatos NF, associados aos eventos prioritários, em série com os contatos NA dos outros eventos em conflito. Isto faz com que a transição associada a um evento só possa disparar se os eventos que possuem prioridade sobre eles não ocorrerem naquele momento.

4.3 Resultados

A planta funcionou perfeitamente, de acordo com as especificações projetadas. O sistema foi testado com as diversas sequências de peças, mostradas na figura 4.22, tendo os supervisores agido sempre da maneira esperada. A implementação prática foi filmada e se encontra disponível na internet:

- Caso 1: http://www.youtube.com/watch?v=KvBSgLWvVZk
- Caso 2: http://www.youtube.com/watch?v=lDbNMGUmm6k
- Caso 3: http://www.youtube.com/watch?v=RSR41RSjuPE
- Caso 4: http://www.youtube.com/watch?v=kxfsJHPFBLc

(a)

(b)

(c)

(d)

Figura 4.22: Sequências: Caso 1 (a); caso 2 (b); caso 3 (c); caso 4 (d).

Capítulo 5

Conclusões e trabalhos futuros

Este trabalho teve como objetivo a modelagem e implementação de um célula de manufatura, utilizando RPIC e aplicando o método proposto por Moreira et al [9]. A grande vantagem deste procedimento é criar um padrão que converte de rede de Petri interpretada para controle em Ladder diretamente, diminuindo a probabilidade de erros humanos durante a modelagem e programação, além de permitir que outros engenheiros possam compreender mais facilmente um determinado projeto.

Em relação ao trabalho anterior [12], houve vários avanços, dentre eles, maior velocidade no tempo de resposta do robô e melhor aproveitamento das entradas do robô. Porém a grande contribuição deste trabalho foi a aplicação do método [9] para controle modular, nunca antes testado numa aplicação prática. Para tanto foram feitos diversos diagramas Ladder, um para cada elemento da célula. O maior problema encontrado neste projeto se deve pela limitação da garra robótica uma vez que o robô não possui nenhum tipo de realimentação, o que não assegura que o robô, de fato, pegou a peça.

Este trabalho pode ainda ser muito desenvolvido. Uma das maiores necessidades é a adição de técnicas de diagnóstico de falhas para contornar problemas como o do robô falhar ao tentar pegar uma peça. Um outro modo de resolver esse problema seria adicionar sensores de pressão nas garras. Dessa forma, o robô poderia saber se há ou não peça na garra. Uma outra ideia seria utilizar um sensor que diferenciasse as cores, para que peças de outros tipos possam ser colocadas. Pode-se também utilizar outro braço robótico, para que ao invés das peças irem para o depósito, elas recomeçassem o ciclo retornando à esteira de fornecimento.

Bibliografia

[1] C. G. Cassandras e S. Lafortune, *Introduction to Discrete Events Systems*, Springer, New York, NY : USA, 2nd edition.

[2] R. David e H. Alla, *Discrete, Continuous, and Hibrid Petri Nets*, Springer, New York, NY : USA, 2005.

[3] T.Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[4] P. J. Ramadge e W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, pp. 206–230, 1987.

[5] B. E. Butler, J. E. R. Cury, K. Rudie e L. Grigorov, "Conceptual design of discrete-event systems using templates," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 21, pp. 257–303, 2011.

[6] P. J. Ramadge e W. M. Wonham, "Modular supervisory control of discrete event systems," *Mathematics of control of discrete event systems*, vol. 1, pp. 13–30, 1988.

[7] M. H. Queiroz, E. A. P. Santos e J. E. R. Cury, "Síntese modular do controle supervisório em diagrama escada para uma célula de manufatura," in *Anais do V Simpósio Brasileiro de Automação Inteligente*, 2001, vol. 5, pp. 1072–1077.

[8] M. H. Queiroz e J. E. R. Cury, "Controle supervisório modular de sistemas de manufatura," *Revista de Controle e Automação*, vol. 13, pp. 123–133, 2002.

[9] M. V. Moreira, D. S. Botelho e J. C. Basilio, "Ladder diagram implementation of control interpreted petri nets: a state equation approach," in *Proceedings of the 4th IFAC Workshop on Discrete-Event System Design*, 2009, pp. 85–90.

[10] D. P. Fessler, "Sistema de automação de uma máquina injetora de plástico," Projeto final de graduação, UFRJ, Escola Politécnica, 2010.

[11] D. S. Botelho, "Projeto de um sistema de automação de uma célula de manufatura," Projeto final de graduação, UFRJ, Escola Politécnica, 2011.

[12] A. B. Medeiros, "Modelagem e implementação de uma célula de manufatura industrial utilizando a teoria de controle supervisório," Projeto final de graduação, UFRJ, Escola Politécnica, 2009.

[13] K. John e M. Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems*, Springer, New York, NY : USA, 2005.

[14] Siemens, *S7-1200 Programmable Controller Manual*, Siemens, 2009.

Apêndice A

Rotinas de Operação do Robô

A.1 Operação 1

Step	Speed	XPos	YPos	ZPos	Distance	Base	Shoulder	Elbow	Wrist	Grip
1	80	21,25	12,53	8,92	23,04	-22,78	-43,2	66,9	-90	-90
2	100	21,06	12,5	8,75	22,81	-22,56	-42	64,92	-90	-90
3	10	21,39	9,77	8,99	23,2	-22,78	-39,4	47,15	-75,03	-90
4	60	22,06	8,04	9,27	23,93	-22,78	-39,4	47,15	-75,03	-20
5	70	20,85	8,94	9,85	23,06	-25,29	-34,2	43,01	-75,84	-20
6	70	15,58	12,21	12,8	20,16	-39,42	-19,6	37,08	-83,12	-20
7	70	12,04	13,28	16,57	20,49	-54	-23,8	48,34	-90	-20
8	80	0,35	13,59	14,6	14,6	-88,63	6,8	14,18	-85,75	-20
9	70	0,41	9	17,24	17,25	-88,63	-2	0,36	-63,1	-20
10	20	0,41	6,24	17,24	17,24	-88,63	-3,8	-11,1	-49,75	-20
11	20	0,41	10,48	17,35	17,36	-88,63	-6,8	4,7	-63,3	-90
12	65	0,33	11,19	13,67	13,67	-88,63	13,6	-3,59	-75,24	-22,8

A.2 Operação 2

Step	Speed	XPos	YPos	ZPos	Distance	Base	Shoulder	Elbow	Wrist	Grip
1	80	22,14	10,27	8,89	23,86	-21,87	-41,8	61,76	-86,36	-20
2	80	21,44	12,04	8,61	23,11	-21,87	-42	62,36	-86,76	-90
3	15	21,7	9,8	8,41	23,7	-21,19	-39,2	46,76	-74,22	-90
4	25	22,62	8,1	8,67	24,23	-20,96	-40,8	49,52	-75,44	-20
5	85	20,86	9,81	9,86	23,07	-25,29	-35,6	49,52	-81,1	-20
6	85	17	13	5,37	17,83	-17,54	-11	30,96	-87,98	-20
7	85	3,13	13,41	-15,23	15,55	78,38	6,6	11,61	-78,88	-20
8	85	0	9,41	-16,74	16,74	90	-0,6	1,34	-66,74	-20
9	25	0	8,88	-15,9	15,9	90	-1,2	-8,73	-56,22	-90
10	25	0	10,41	-17	17	90	-2,4	8,06	-72	-20

A.3 Operação 3

Step	Speed	XPos	YPos	ZPos	Distance	Base	Shoulder	Elbow	Wrist	Grip
1	75	2,19	10,45	17,71	17,84	-82,94	-5,8	11,61	-71,39	-20
2	15	0,97	10,11	17,45	17,48	-86,81	-7,6	3,91	-61,89	-90
3	15	0,97	9,19	17,43	17,46	-86,81	-8	-0,04	-57,64	-90
4	15	2,26	7,8	18,26	18,4	-82,94	-8,8	2,53	-59,66	-20
5	60	0,8	11,32	14,43	14,45	-86,81	9,6	0,95	-76,25	-20
6	90	0	11,28	-14,46	14,46	90	9,6	0,75	-76,04	-20
7	50	0	9,43	-16,77	16,77	90	-0,6	1,34	-66,54	-20
8	40	0	7,16	-16,66	16,66	90	-1,2	-8,73	-56,22	-20
9	75	0	8,85	-15,84	15,84	90	-1,2	-8,73	-56,63	-90
10	90	0	10,44	-17,03	17,03	90	-2,2	7,86	-71,6	-20

A.4 Operação 4

Step	Speed	XPos	YPos	ZPos	Distance	Base	Shoulder	Elbow	Wrist	Grip
1	75	1,95	10,45	-16,83	16,94	83,39	-1,8	7,46	-71,6	-20
2	25	3,67	9,11	-16,33	17,22	77,7	-3,6	3,32	-66,54	-20
3	90	3,51	10,67	-16,11	16,48	77,7	-3,6	2,53	-65,73	-90
4	15	4,27	9,45	-15,91	16,47	74,96	-4	-2,8	-60,07	-90
5	80	3,29	7,53	-17,02	17,33	79,06	-4,2	-3,79	-58,25	-20
6	70	1,98	12,66	-17,07	17,19	83,39	-5,6	22,47	-83,33	-20
7	75	15,58	14,47	-12,29	19,84	38,28	-8,8	28,79	-73,42	-20
8	70	17,38	14,44	9,52	19,82	-28,71	-8,6	28,39	-73,21	-20
9	65	20,17	12,04	15,53	25,46	-37,59	-38,2	58,21	-73,42	-20
10	15	20,45	10,11	17,64	27	-40,78	-46,6	62,95	-69,78	-20
11	70	19,57	11,6	16,89	25,85	-40,78	-46,2	62,16	-69,37	-90
12	75	21,27	12,66	11,65	24,25	-28,71	-35,4	49,32	-67,35	-85,1

Apêndice B

Programação das Entradas do Robô

Do

If Input #1
If Input #3
Else
If Input #4
Else
0001
EndIf
EndIf
EndIf

If Input #3
If Input #1
Else
If Input #4
Else
0002
EndIf
EndIf
EndIf

If Input #4
If Input #1
Else
If Input #3
Else
0003
EndIf
EndIf
EndIf

If Input #1
If Input #3
If Input #4
Else
0004
EndIf
EndIf
EndIf

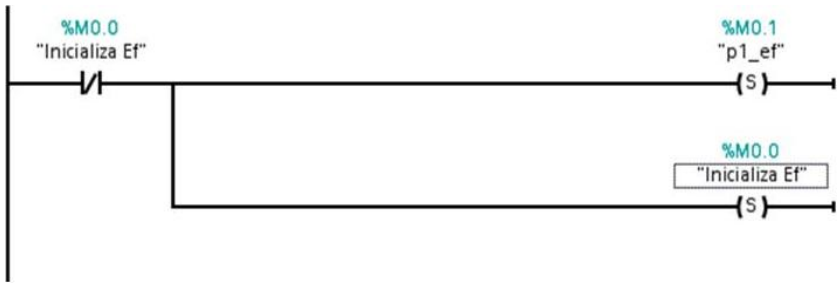
While Input #2

Apêndice C

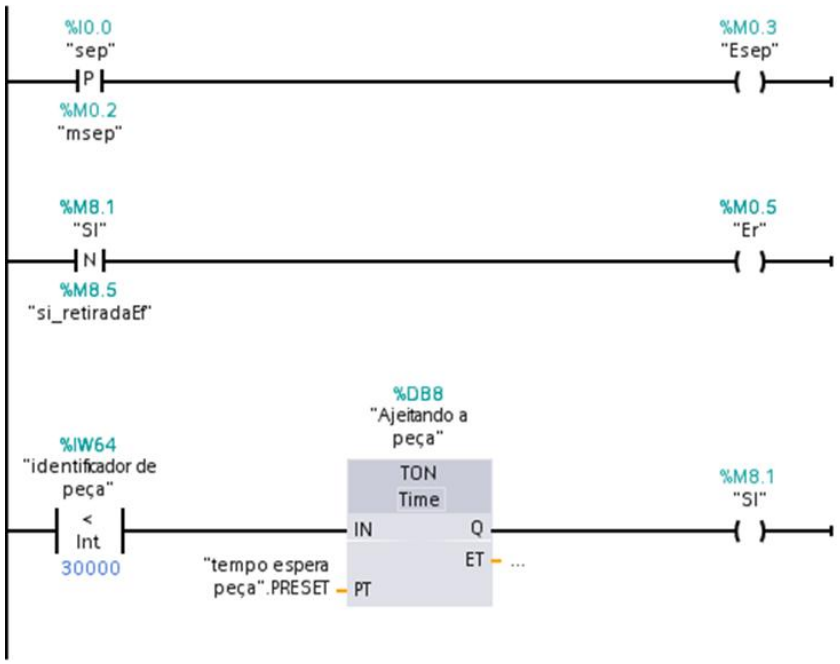
Arquivos em Ladder

C.1 Esteira de Fornecimento

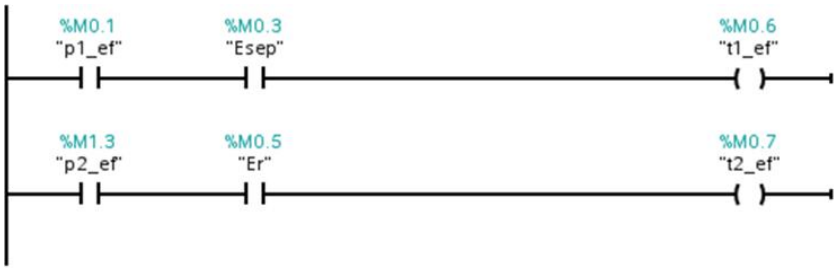
Módulo de Inicialização



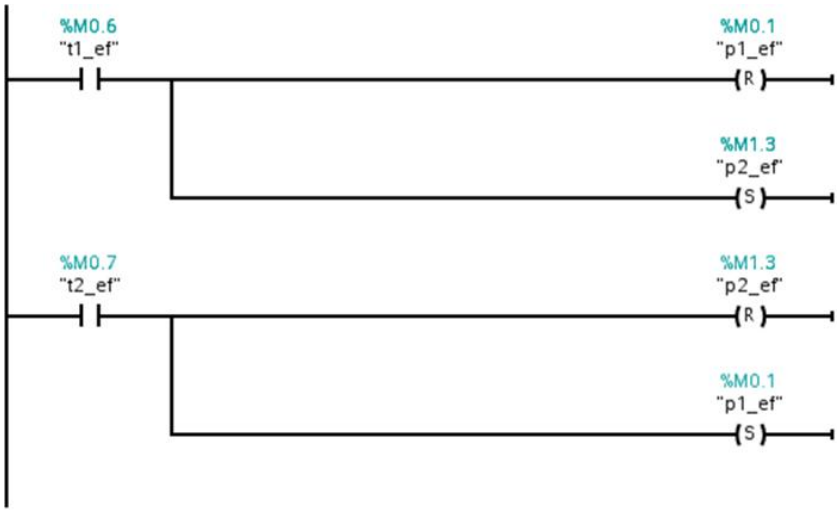
Módulo de Eventos



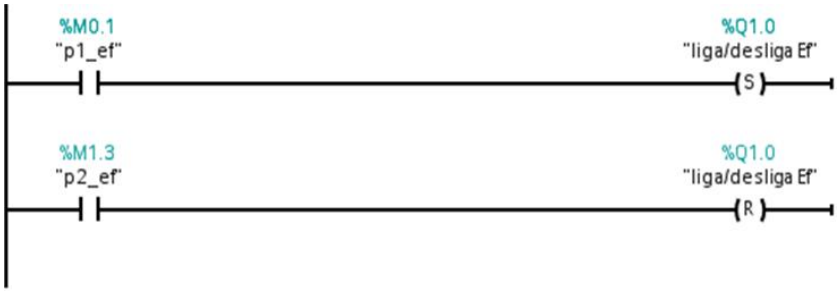
Módulo de Condições para o Disparo



Módulo de Dinâmica

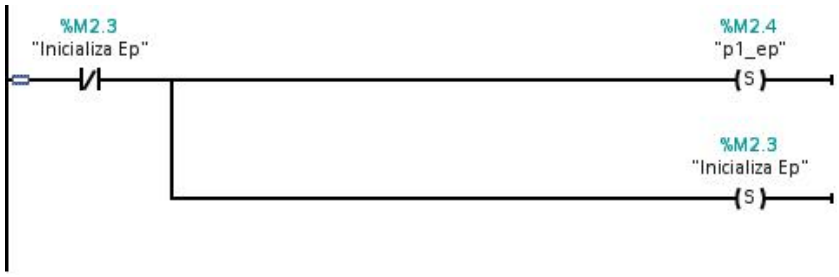


Módulo de Ações

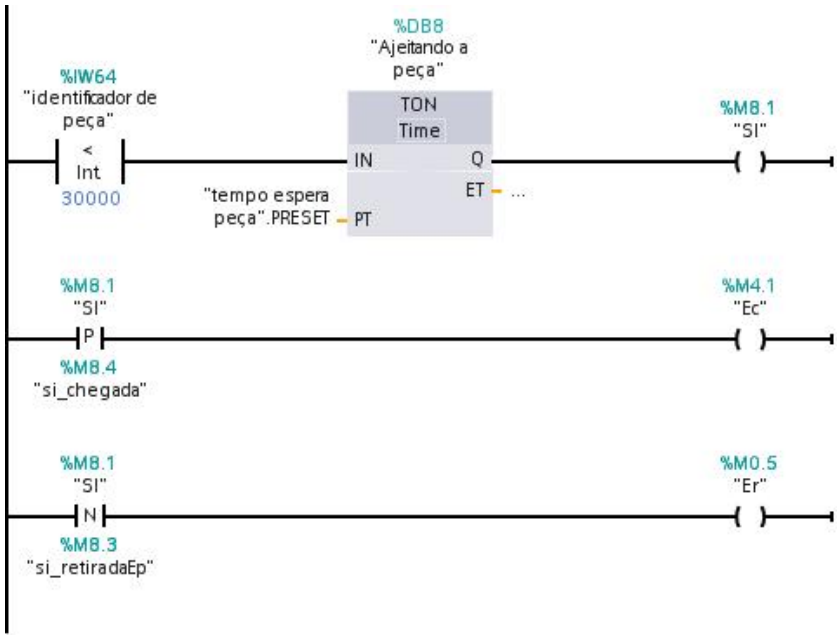


C.2 Esteira Principal

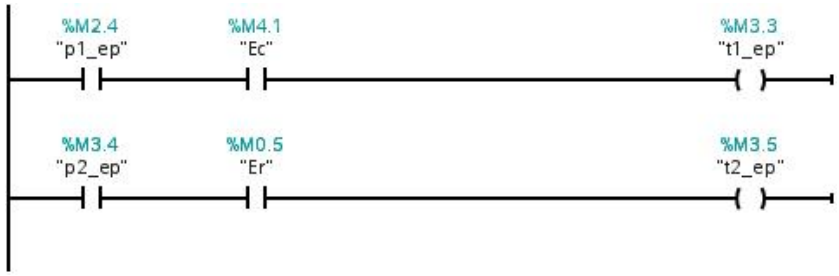
Módulo de Inicialização



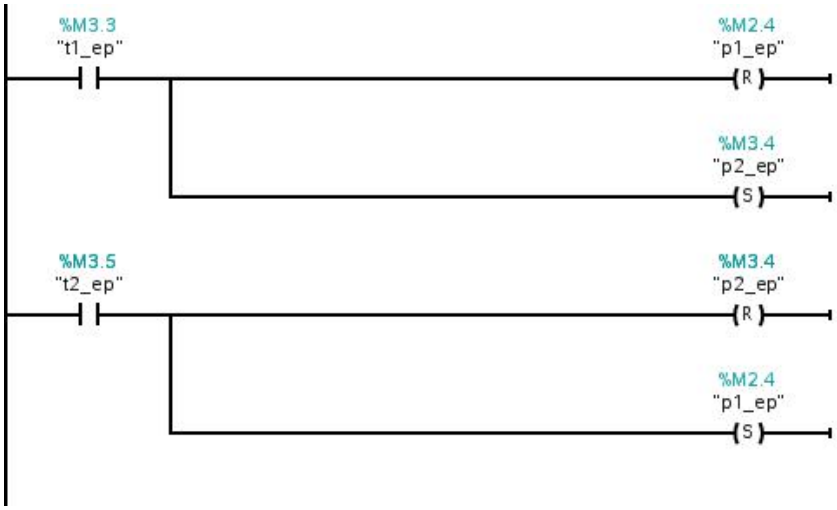
Módulo de Eventos



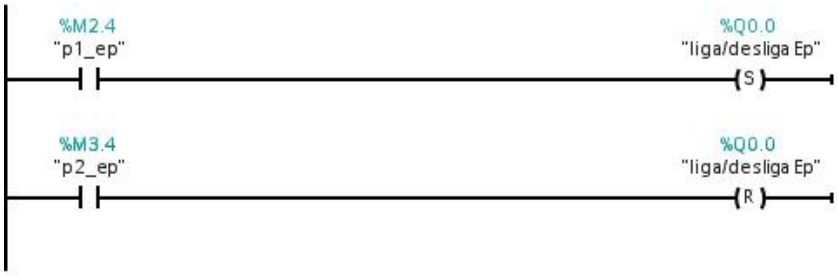
Módulo de Condições para o Disparo



Módulo de Dinâmica

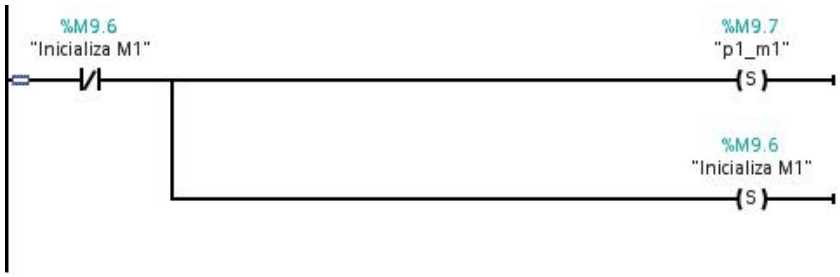


Módulo de Ações

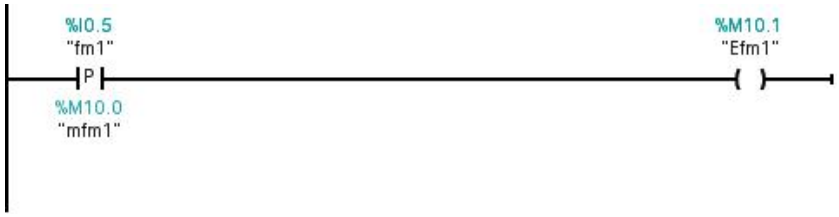


C.3 Máquina M1

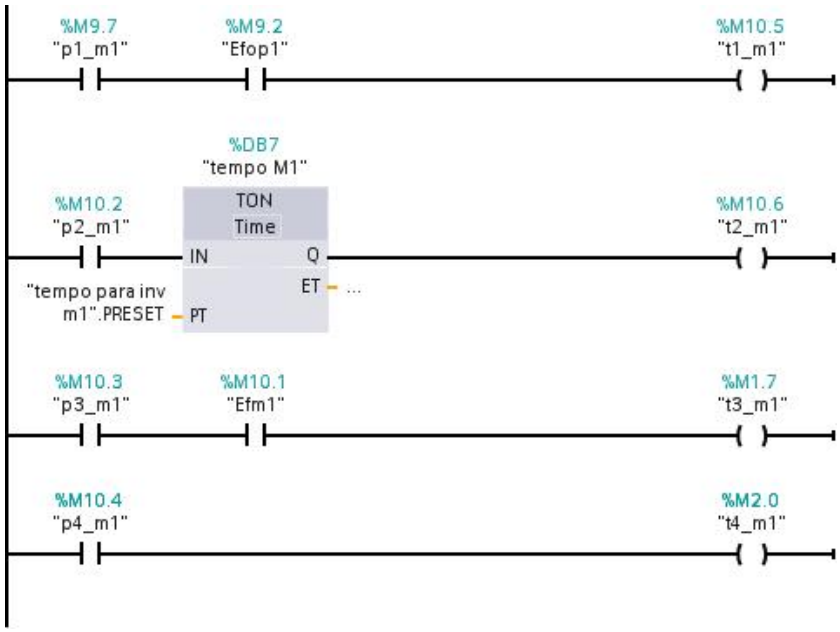
Módulo de Inicialização



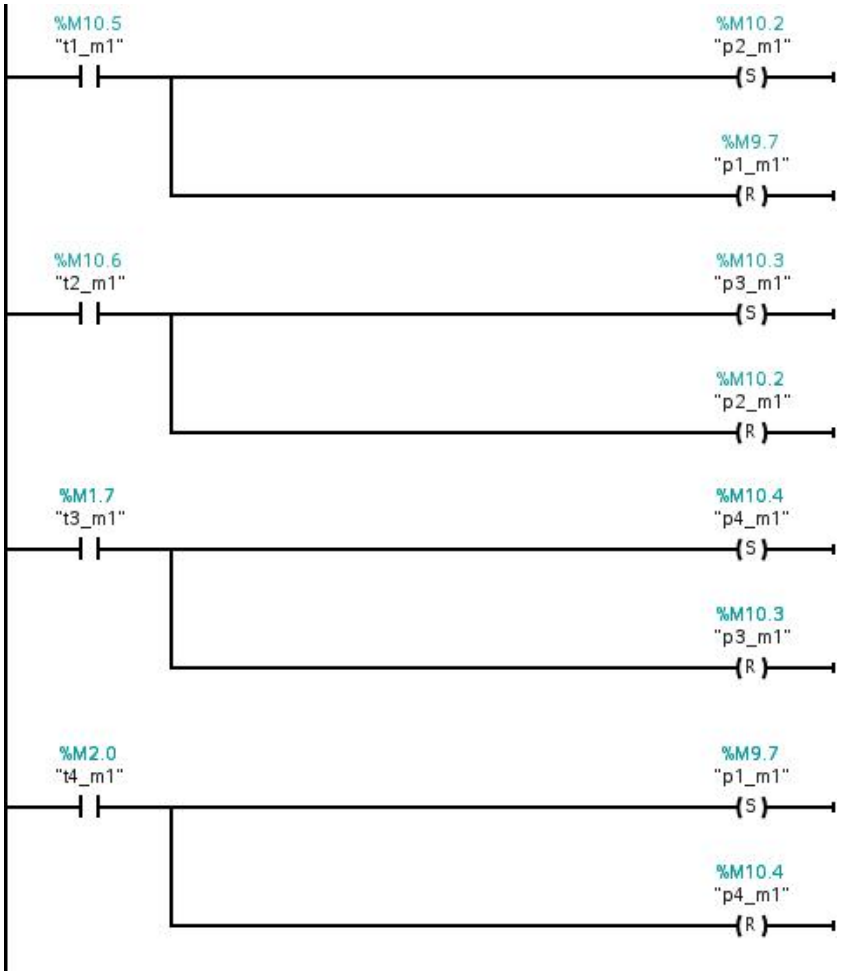
Módulo de Eventos



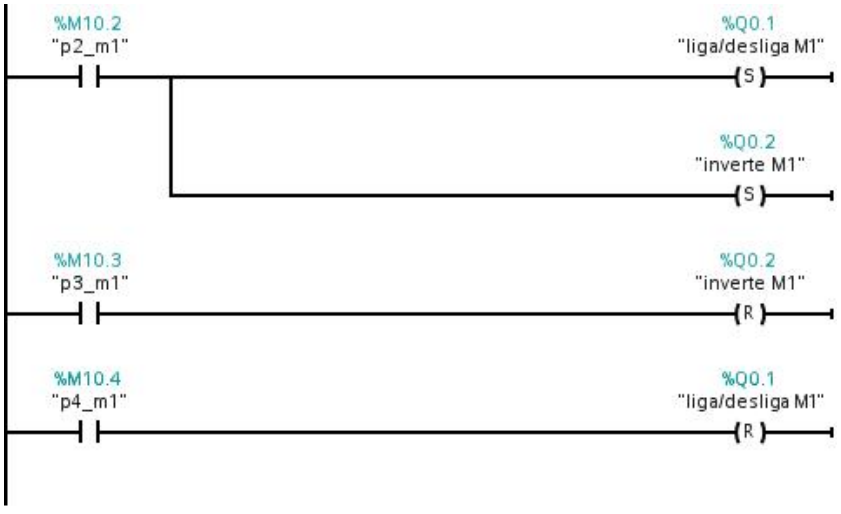
Módulo de Condições para o Disparo



Módulo de Dinâmica

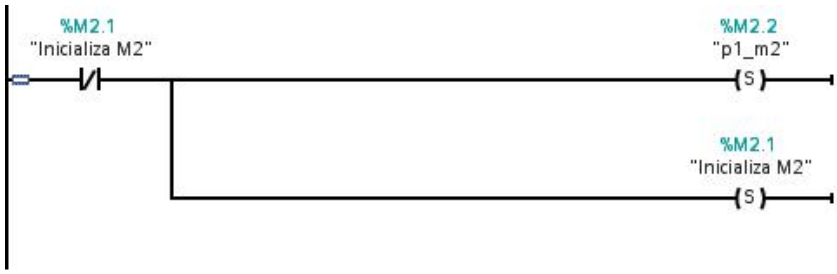


Módulo de Ações

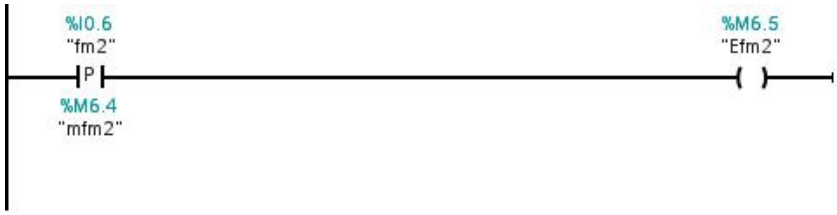


C.4 Máquina M2

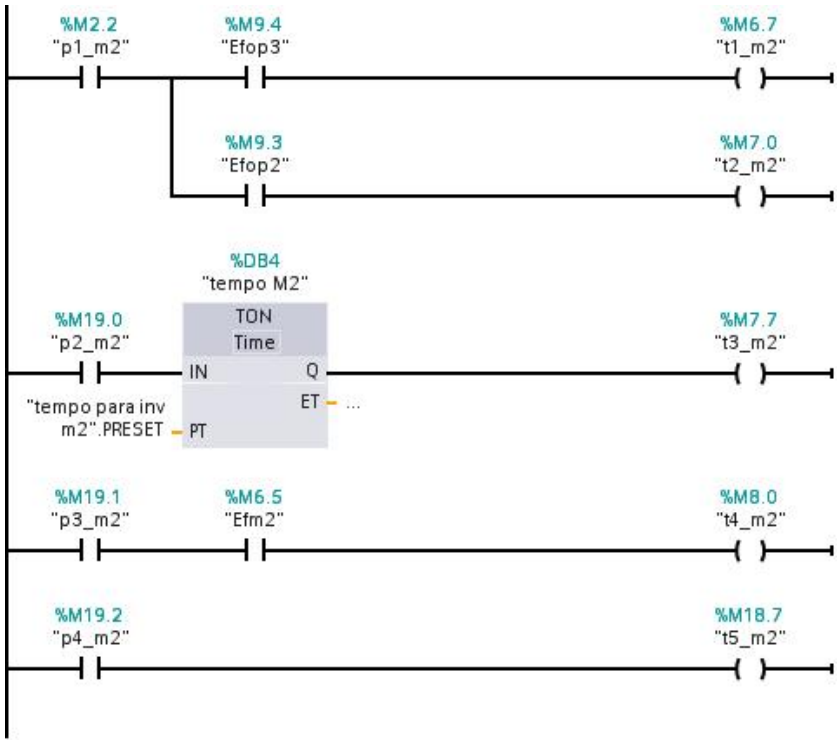
Módulo de Inicialização



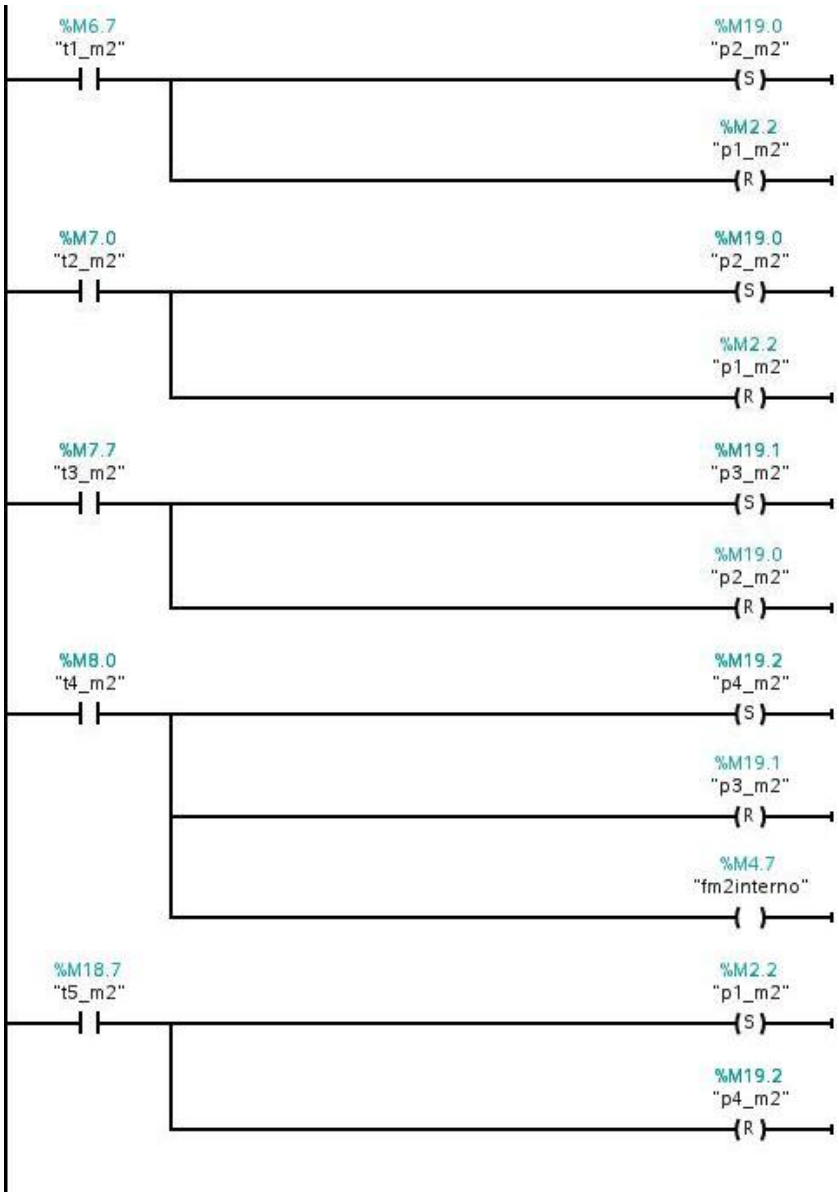
Módulo de Eventos



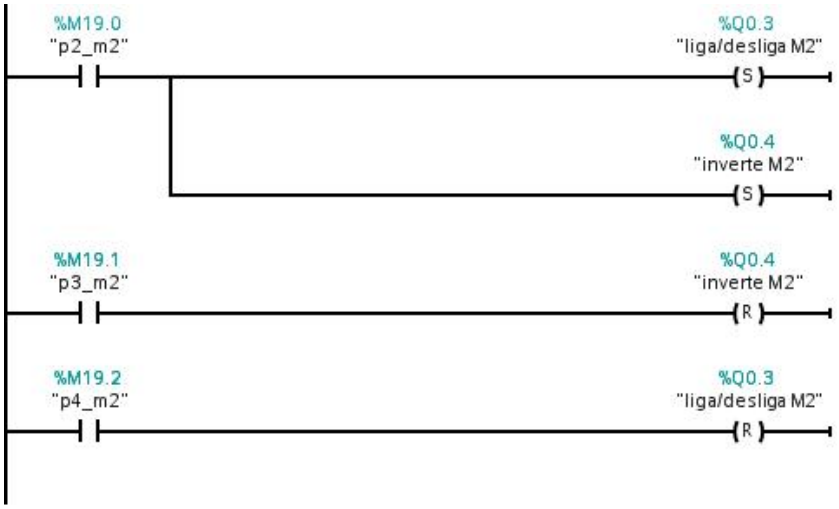
Módulo de Condições para o Disparo



Módulo de Dinâmica

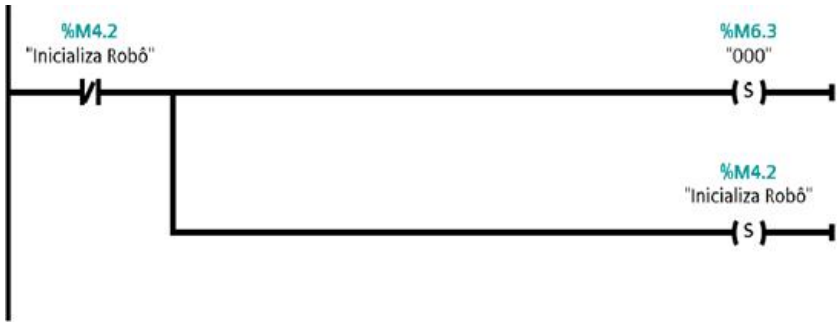


Módulo de Ações

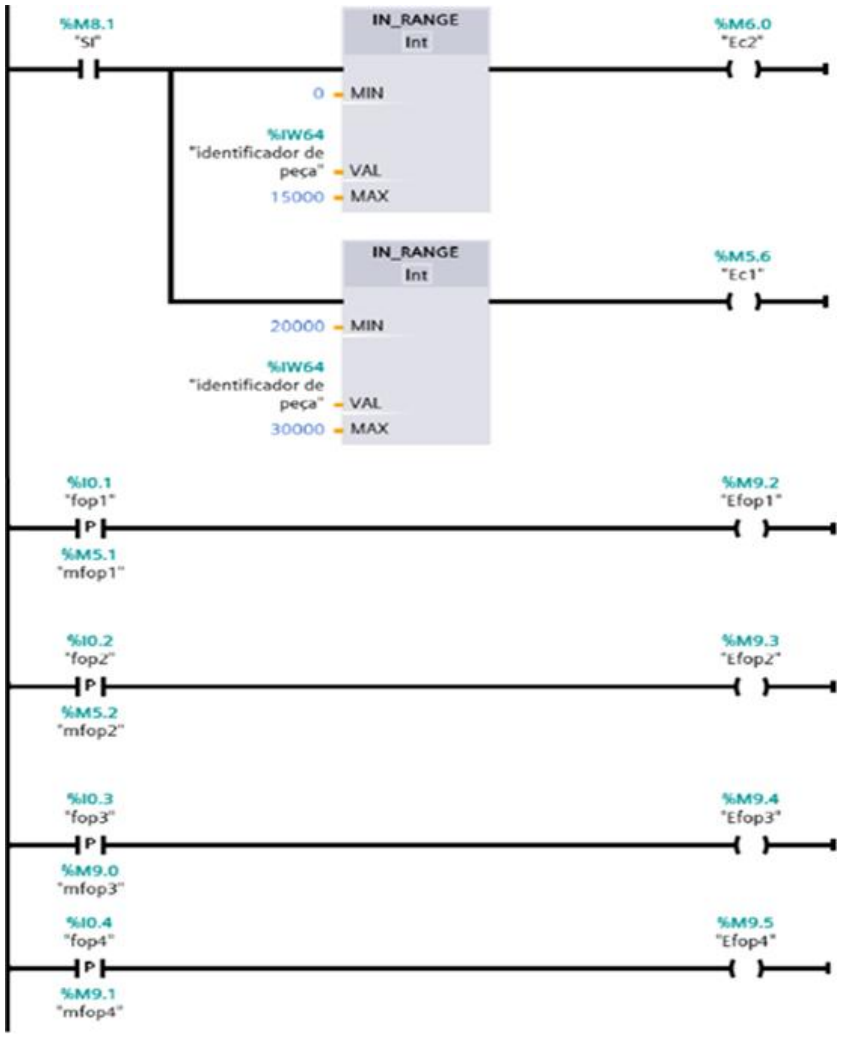


C.5 Robô

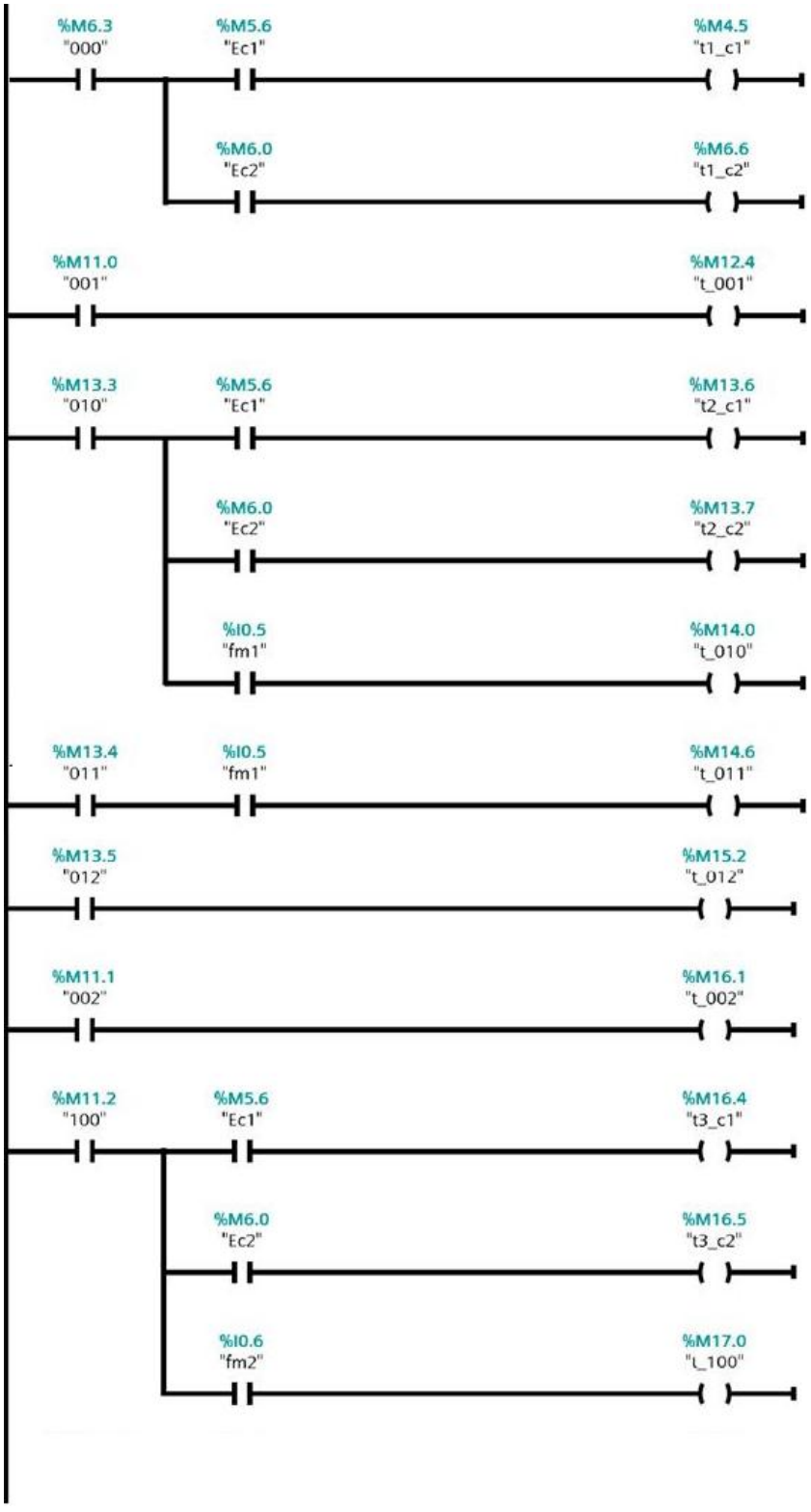
Módulo de Inicialização

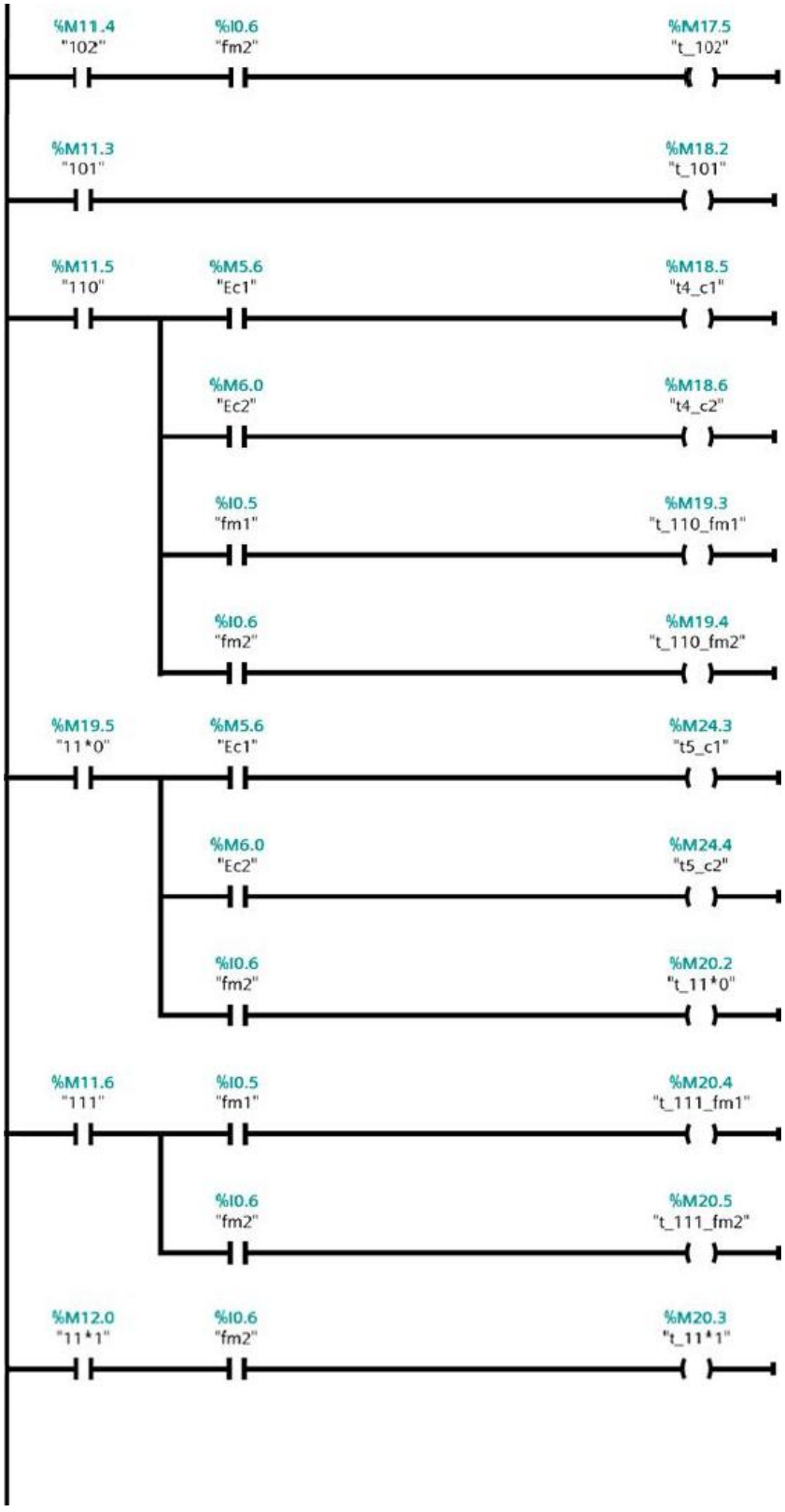


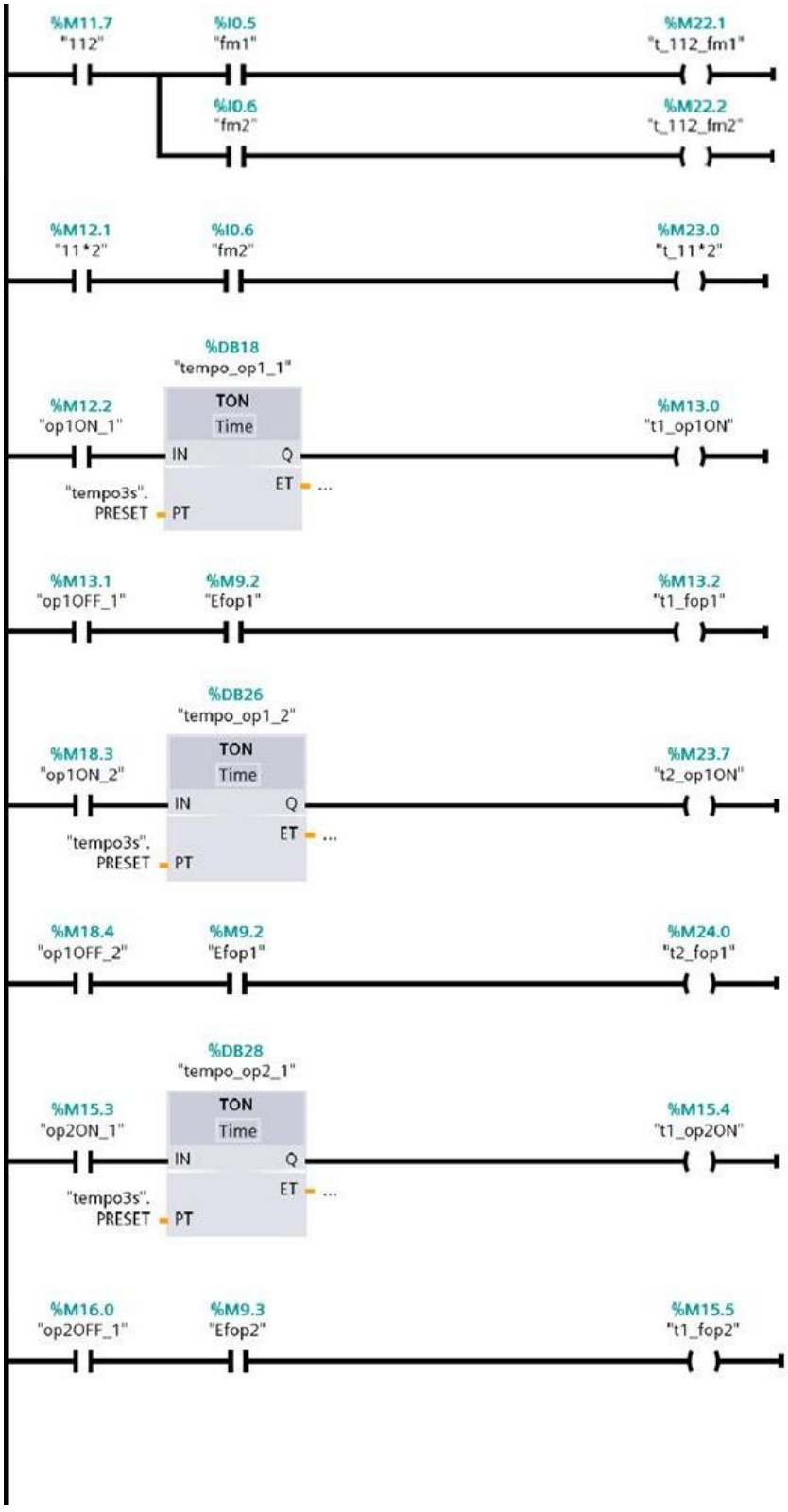
Módulo de Eventos

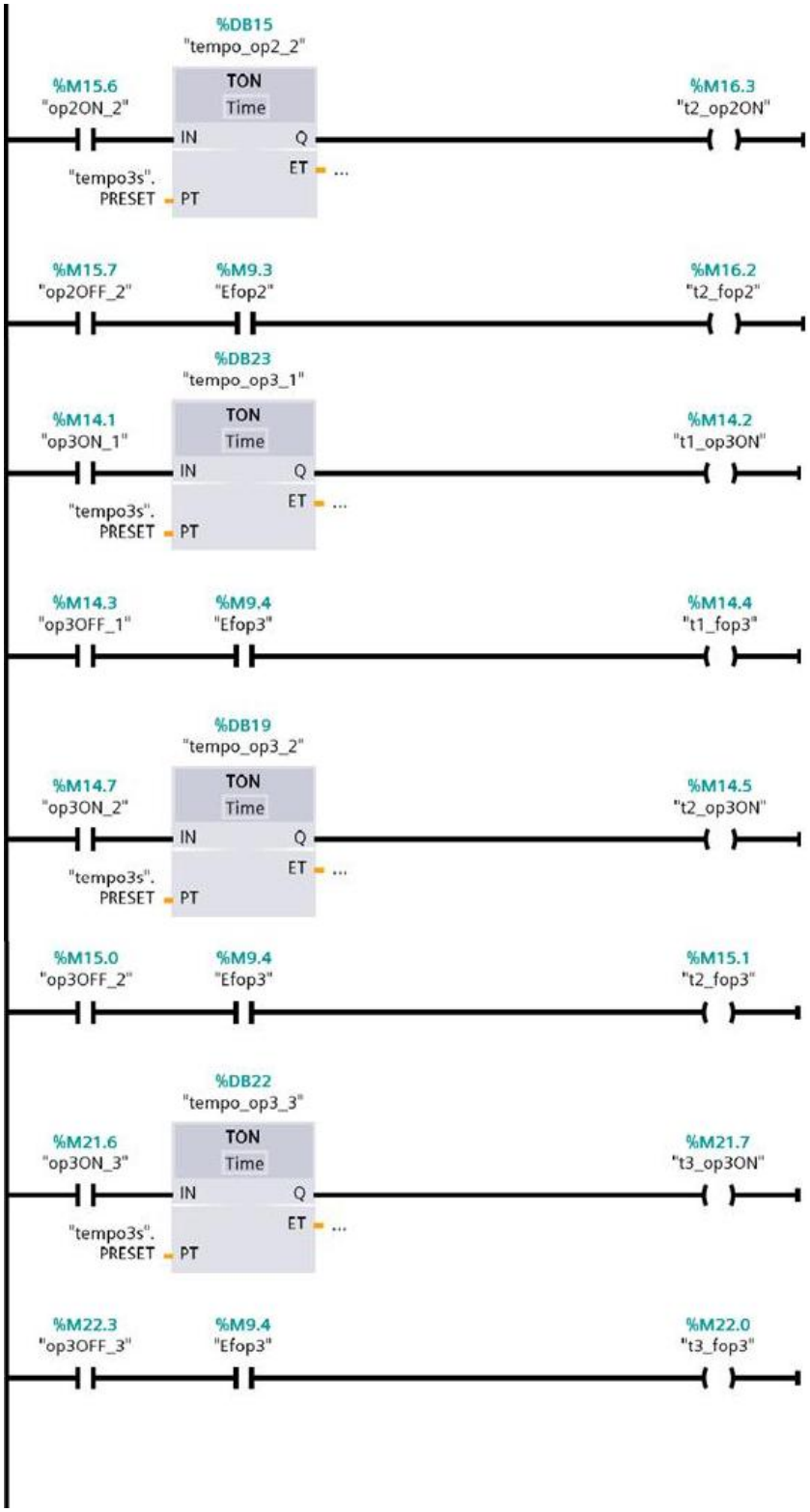


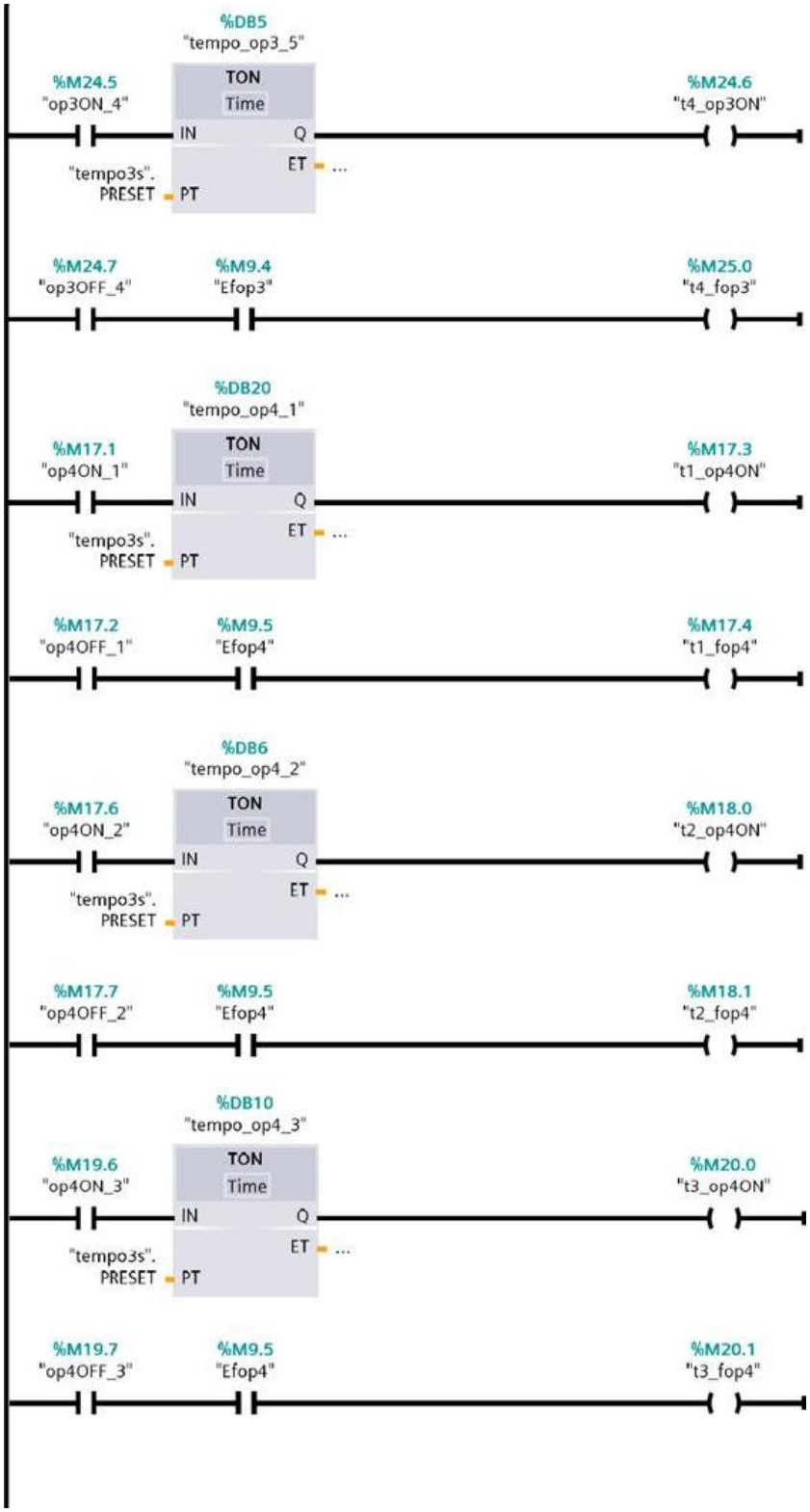
Módulo de Condições para o Disparo

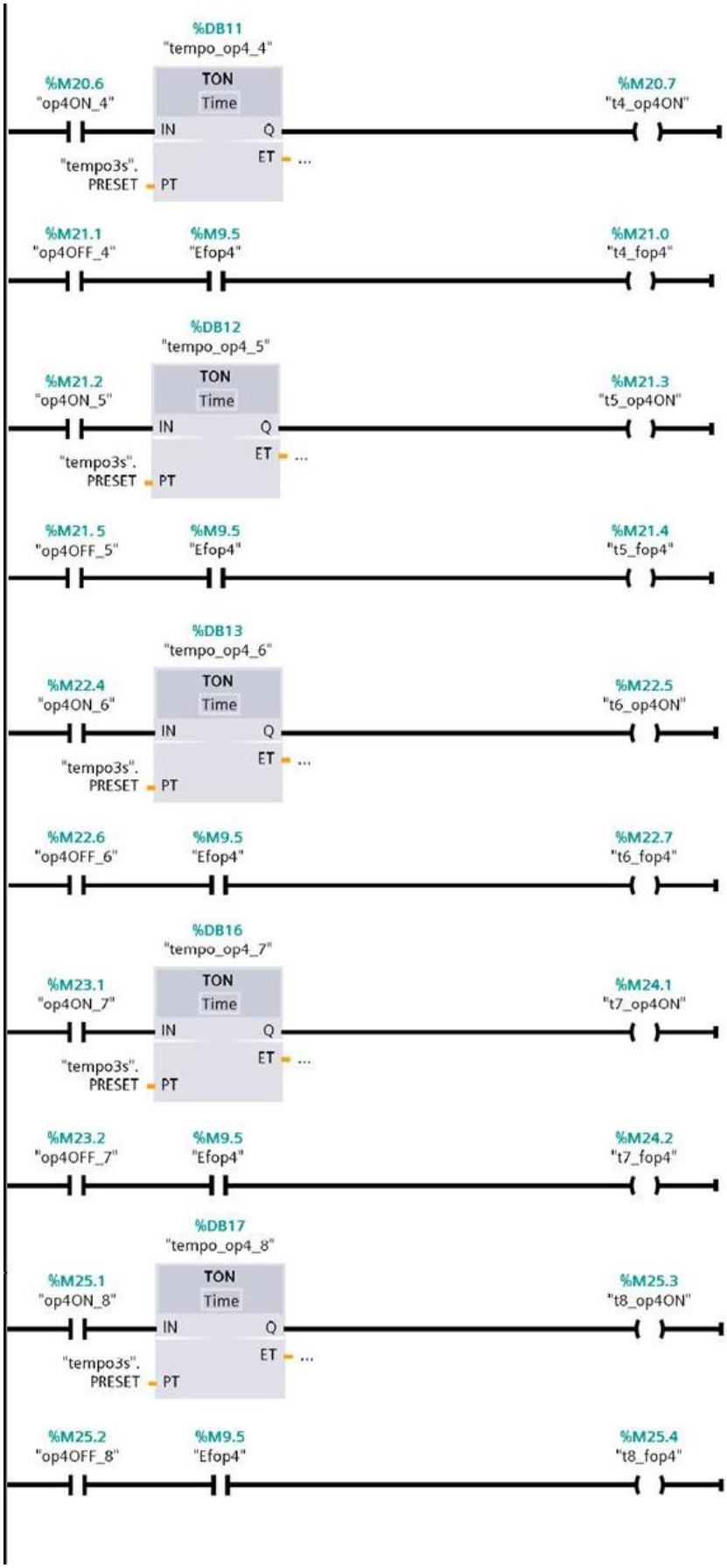



%M11.4
"102"
%I0.6
"fm2"
%M17.5
"t_102"
%M11.3
"101"
%M18.2
"t_101"
%M11.5
"110"
%M5.6
"Ec1"
%M18.5
"t4_c1"
%M6.0
"Ec2"
%M18.6
"t4_c2"
%I0.5
"fm1"
%M19.3
"t_110_fm1"
%I0.6
"fm2"
%M19.4
"t_110_fm2"
%M19.5
"11*0"
%M5.6
"Ec1"
%M24.3
"t5_c1"
%M6.0
"Ec2"
%M24.4
"t5_c2"
%I0.6
"fm2"
%M20.2
"t_11*0"
%M11.6
"111"
%I0.5
"fm1"
%M20.4
"t_111_fm1"
%I0.6
"fm2"
%M20.5
"t_111_fm2"
%M12.0
"11*1"
%I0.6
"fm2"
%M20.3
"t_11*1"

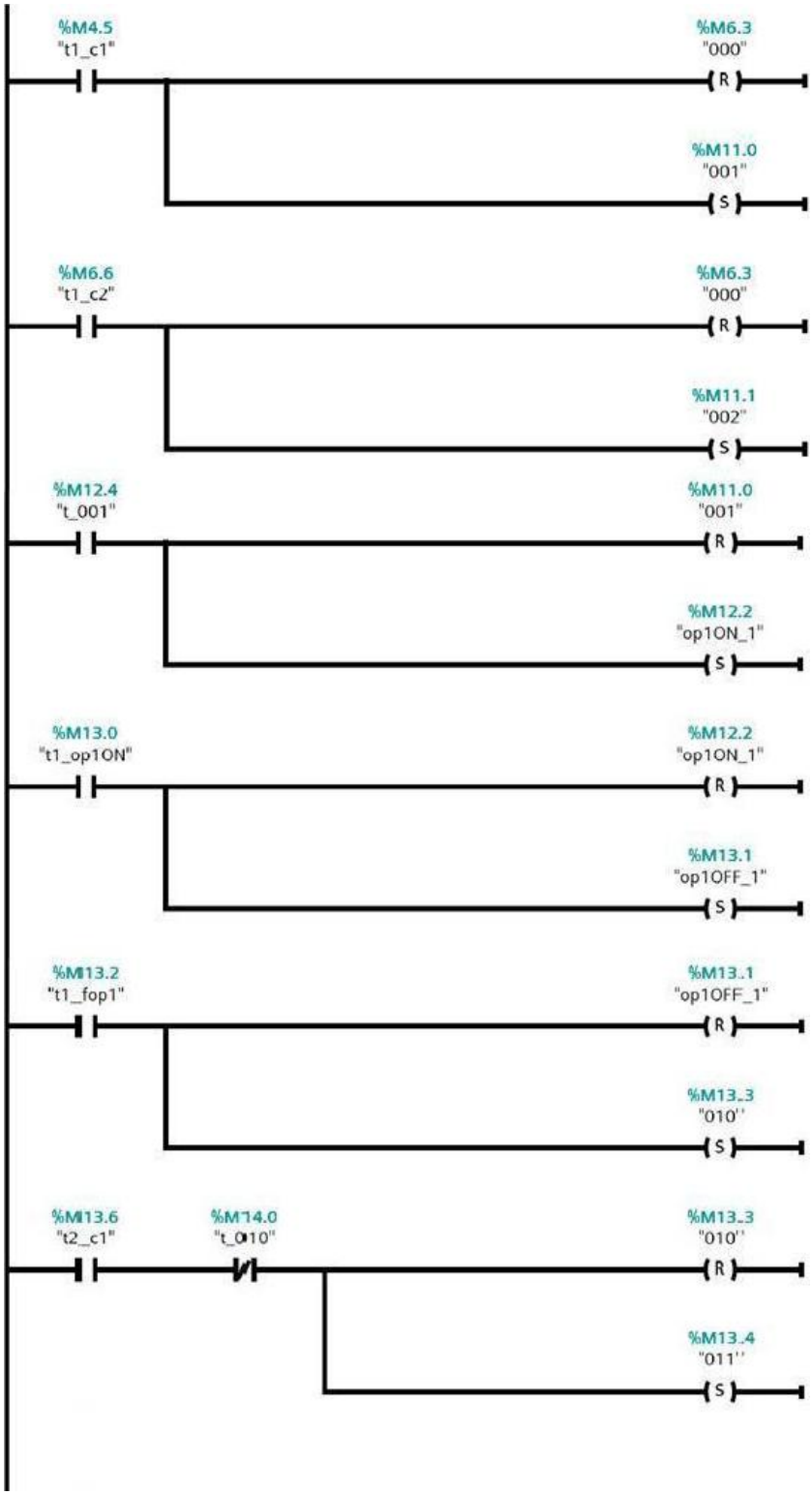

%M11.7
"112"
%I0.5
"fm1"
%M22.1
"t_112_fm1"
%I0.6
"fm2"
%M22.2
"t_112_fm2"
%M12.1
"11*2"
%I0.6
"fm2"
%M23.0
"t_11*2"
%DB18
"tempo_op1_1"
%M12.2
"op1ON_1"
TON
Time
IN
Q
ET ...
"tempo3s".
PRESET PT
%M13.0
"t1_op1ON"
%M13.1
"op1OFF_1"
%M9.2
"Efop1"
%M13.2
"t1_fop1"
%DB26
"tempo_op1_2"
%M18.3
"op1ON_2"
TON
Time
IN
Q
ET ...
"tempo3s".
PRESET PT
%M23.7
"t2_op1ON"
%M18.4
"op1OFF_2"
%M9.2
"Efop1"
%M24.0
"t2_fop1"
%DB28
"tempo_op2_1"
%M15.3
"op2ON_1"
TON
Time
IN
Q
ET ...
"tempo3s".
PRESET PT
%M15.4
"t1_op2ON"
%M16.0
"op2OFF_1"
%M9.3
"Efop2"
%M15.5
"t1_fop2"

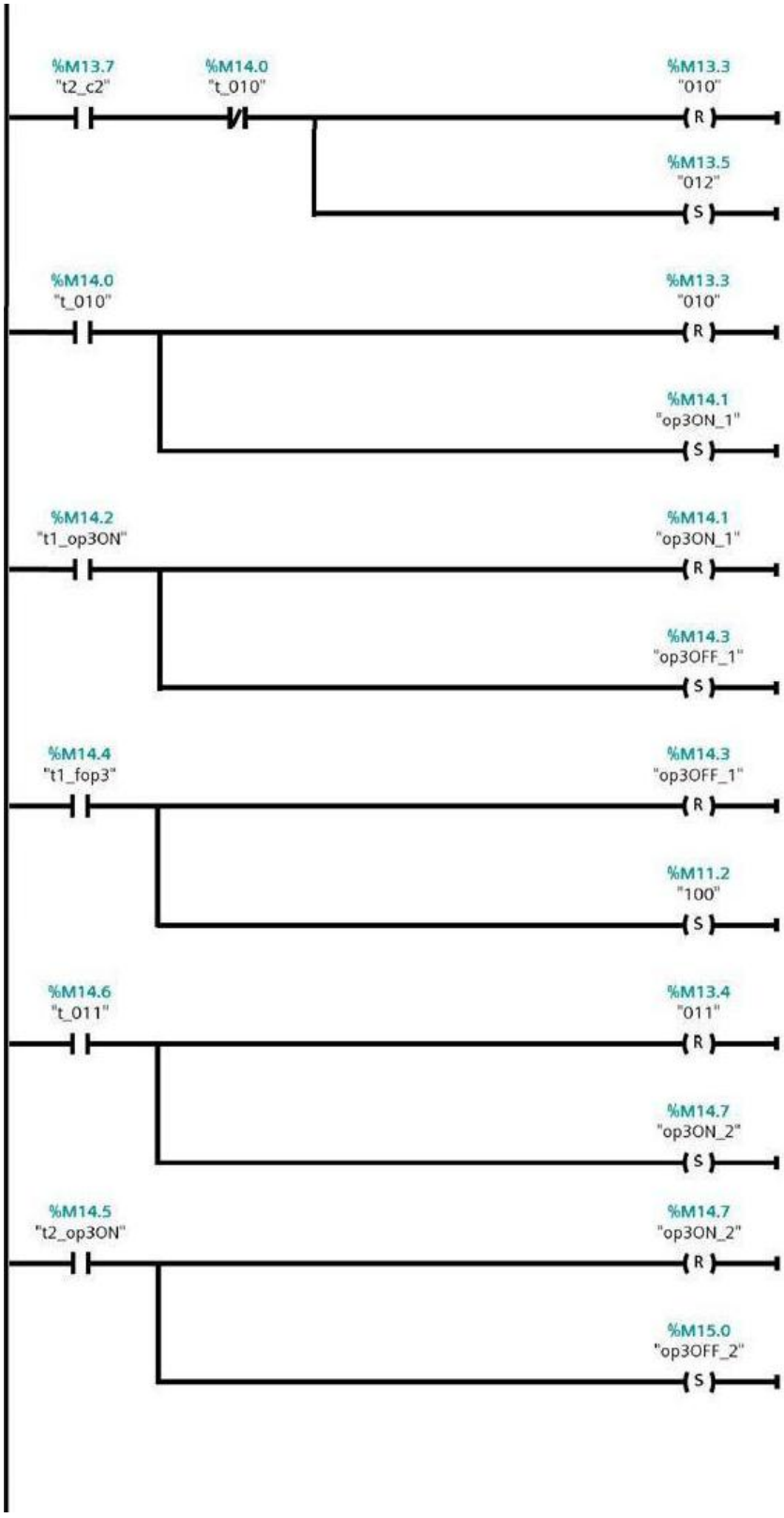

%DB15
"tempo_op2_2"
%M15.6
"op2ON_2"
TON
Time
IN
Q
ET ...
"tempo3s".
PRESET
PT
%M16.3
"t2_op2ON"
%M15.7
"op2OFF_2"
%M9.3
"Efop2"
%M16.2
"t2_fop2"
%DB23
"tempo_op3_1"
%M14.1
"op3ON_1"
%M14.2
"t1_op3ON"
%M14.3
"op3OFF_1"
%M9.4
"Efop3"
%M14.4
"t1_fop3"
%DB19
"tempo_op3_2"
%M14.7
"op3ON_2"
%M14.5
"t2_op3ON"
%M15.0
"op3OFF_2"
%M15.1
"t2_fop3"
%DB22
"tempo_op3_3"
%M21.6
"op3ON_3"
%M21.7
"t3_op3ON"
%M22.3
"op3OFF_3"
%M22.0
"t3_fop3"

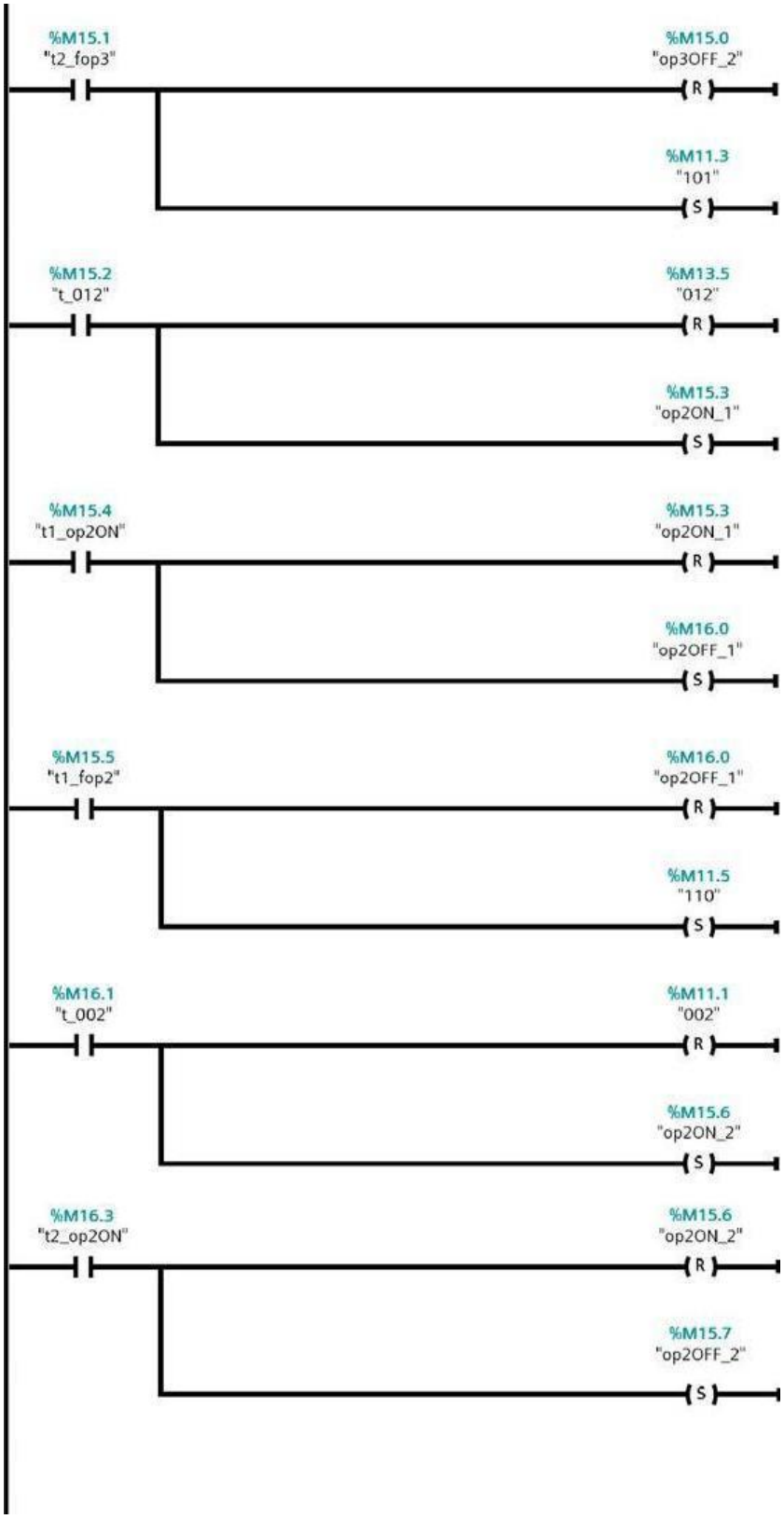

%DB5
"tempo_op3_5"
%M24.5
"op3ON_4"
TON
Time
IN
Q
ET ...
"tempo3s".
PRESET
PT
%M24.6
"t4_op3ON"
%M24.7
"op3OFF_4"
%M9.4
"Efop3"
%M25.0
"t4_fop3"
%DB20
"tempo_op4_1"
%M17.1
"op4ON_1"
%M17.3
"t1_op4ON"
%M17.2
"op4OFF_1"
%M9.5
"Efop4"
%M17.4
"t1_fop4"
%DB6
"tempo_op4_2"
%M17.6
"op4ON_2"
%M18.0
"t2_op4ON"
%M17.7
"op4OFF_2"
%M18.1
"t2_fop4"
%DB10
"tempo_op4_3"
%M19.6
"op4ON_3"
%M20.0
"t3_op4ON"
%M19.7
"op4OFF_3"
%M20.1
"t3_fop4"

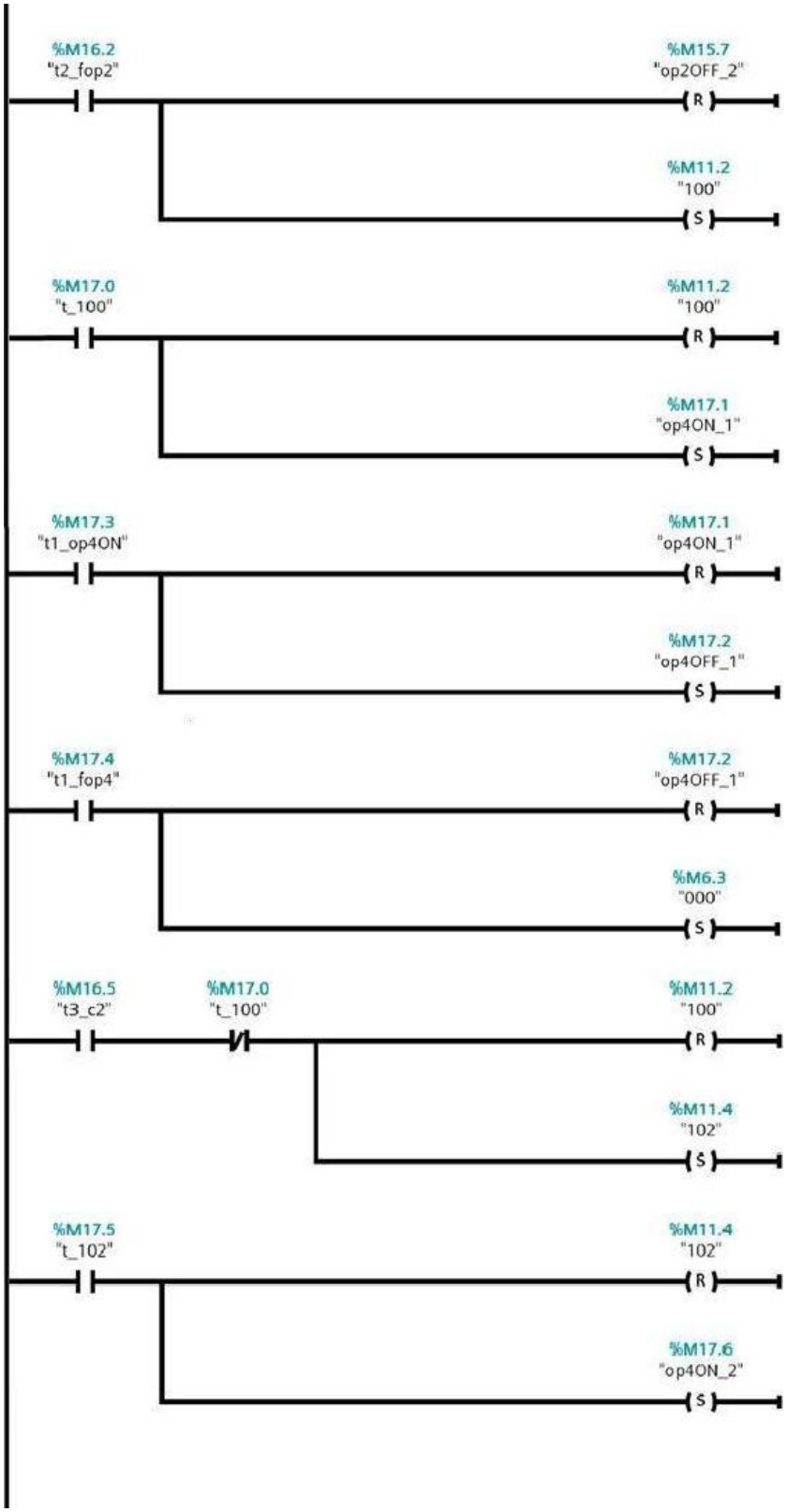

%DB11
"tempo_op4_4"
%M20.6
"op4ON_4"
TON
Time
IN
Q
ET ...
"tempo3s".
PRESET
PT
%M20.7
"t4_op4ON"
%M21.1
"op4OFF_4"
%M9.5
"Efop4"
%M21.0
"t4_fop4"
%DB12
"tempo_op4_5"
%M21.2
"op4ON_5"
%M21.3
"t5_op4ON"
%M21.5
"op4OFF_5"
%M21.4
"t5_fop4"
%DB13
"tempo_op4_6"
%M22.4
"op4ON_6"
%M22.5
"t6_op4ON"
%M22.6
"op4OFF_6"
%M22.7
"t6_fop4"
%DB16
"tempo_op4_7"
%M23.1
"op4ON_7"
%M24.1
"t7_op4ON"
%M23.2
"op4OFF_7"
%M24.2
"t7_fop4"
%DB17
"tempo_op4_8"
%M25.1
"op4ON_8"
%M25.3
"t8_op4ON"
%M25.2
"op4OFF_8"
%M25.4
"t8_fop4"

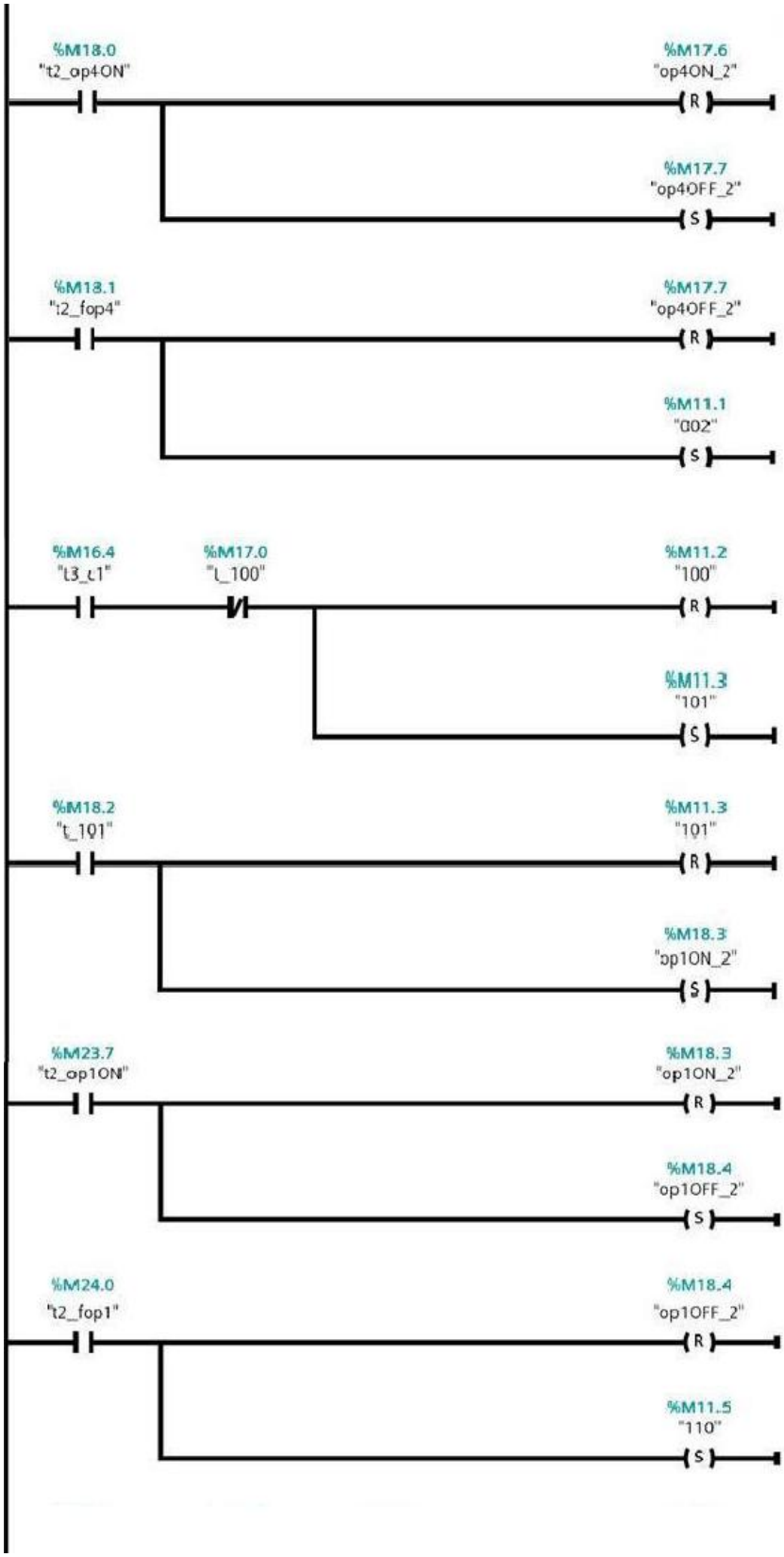
Módulo de Dinâmica

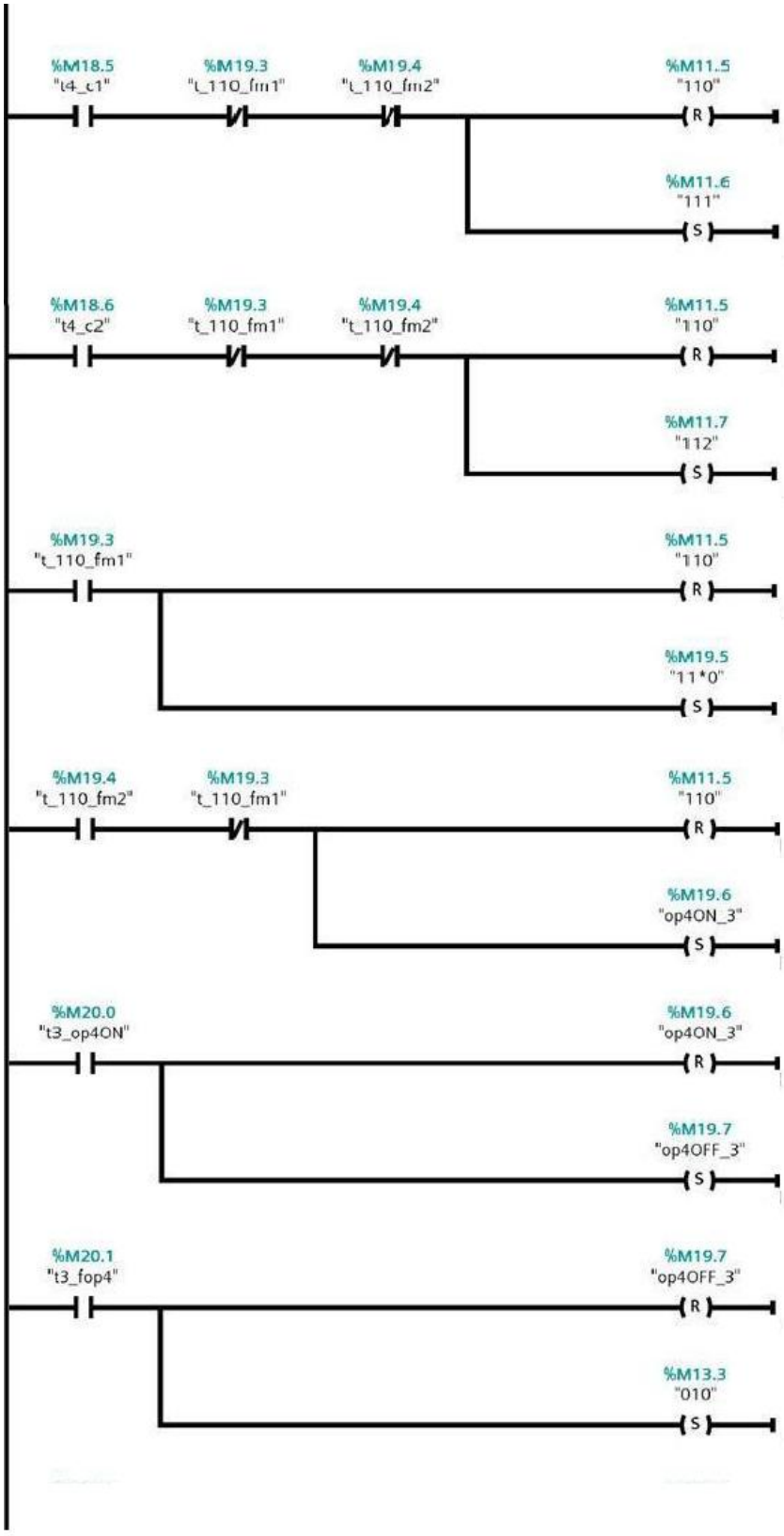


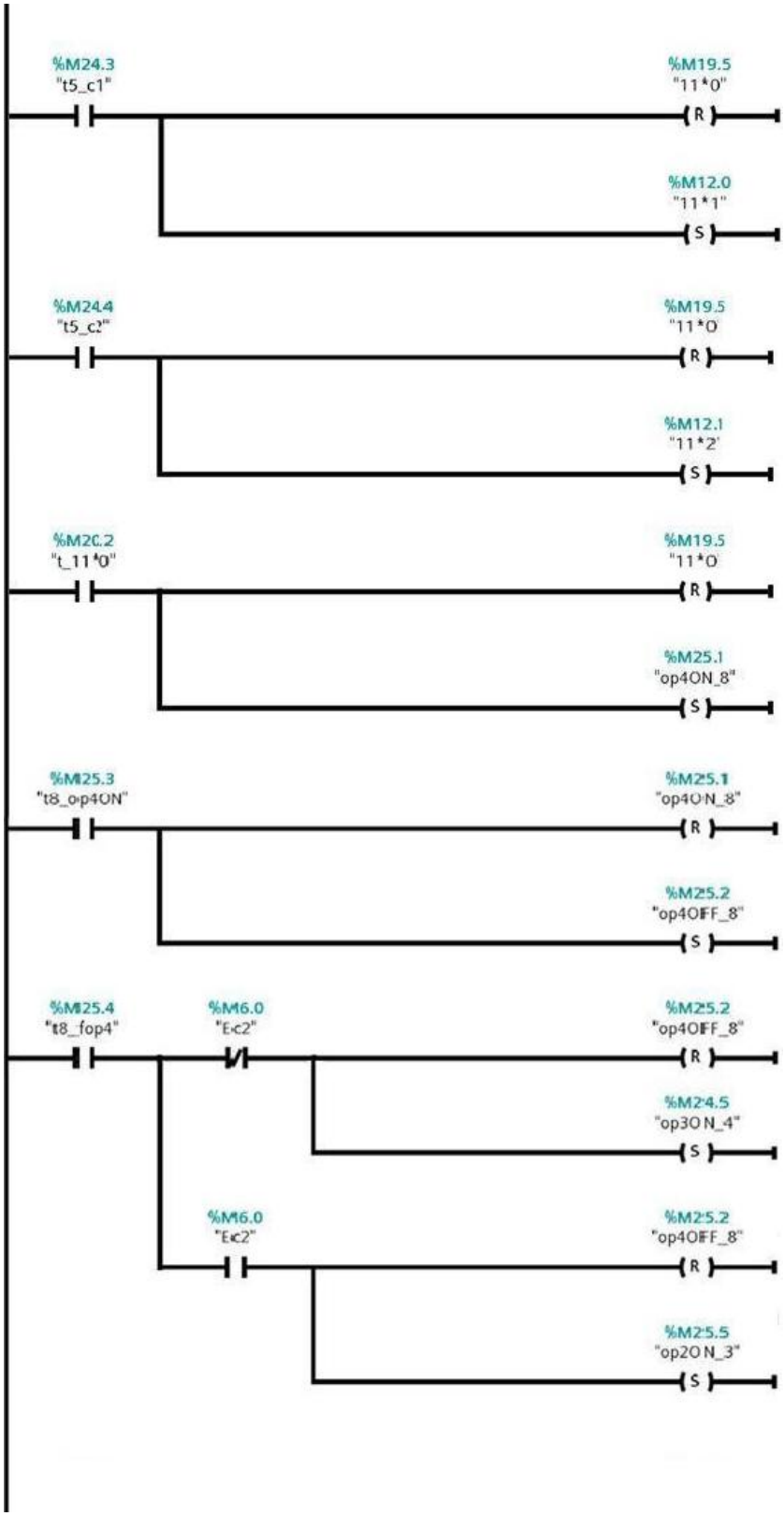

%M13.7
"t2_c2"
%M14.0
"t_010"
%M13.3
"010"
R
%M13.5
"012"
S
%M14.0
"t_010"
%M13.3
"010"
R
%M14.1
"op3ON_1"
S
%M14.2
"t1_op3ON"
%M14.1
"op3ON_1"
R
%M14.3
"op3OFF_1"
S
%M14.4
"t1_fop3"
%M14.3
"op3OFF_1"
R
%M11.2
"100"
S
%M14.6
"t_011"
%M13.4
"011"
R
%M14.7
"op3ON_2"
S
%M14.5
"t2_op3ON"
%M14.7
"op3ON_2"
R
%M15.0
"op3OFF_2"
S

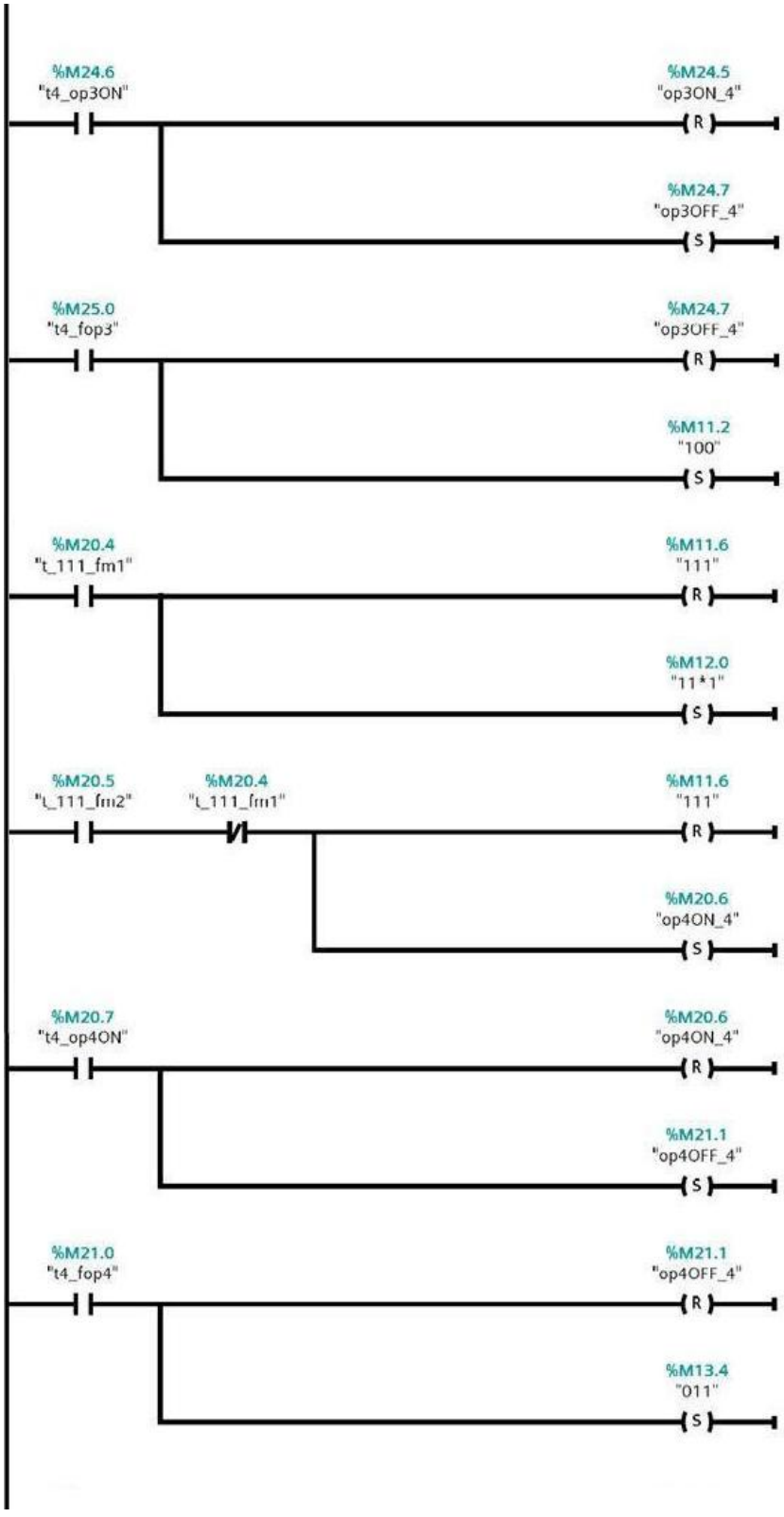

%M15.1
"t2_fop3"
%M15.0
"op3OFF_2"
R
%M11.3
"101"
S
%M15.2
"t_012"
%M13.5
"012"
R
%M15.3
"op2ON_1"
S
%M15.4
"t1_op2ON"
%M15.3
"op2ON_1"
R
%M16.0
"op2OFF_1"
S
%M15.5
"t1_fop2"
%M16.0
"op2OFF_1"
R
%M11.5
"110"
S
%M16.1
"t_002"
%M11.1
"002"
R
%M15.6
"op2ON_2"
S
%M16.3
"t2_op2ON"
%M15.6
"op2ON_2"
R
%M15.7
"op2OFF_2"
S

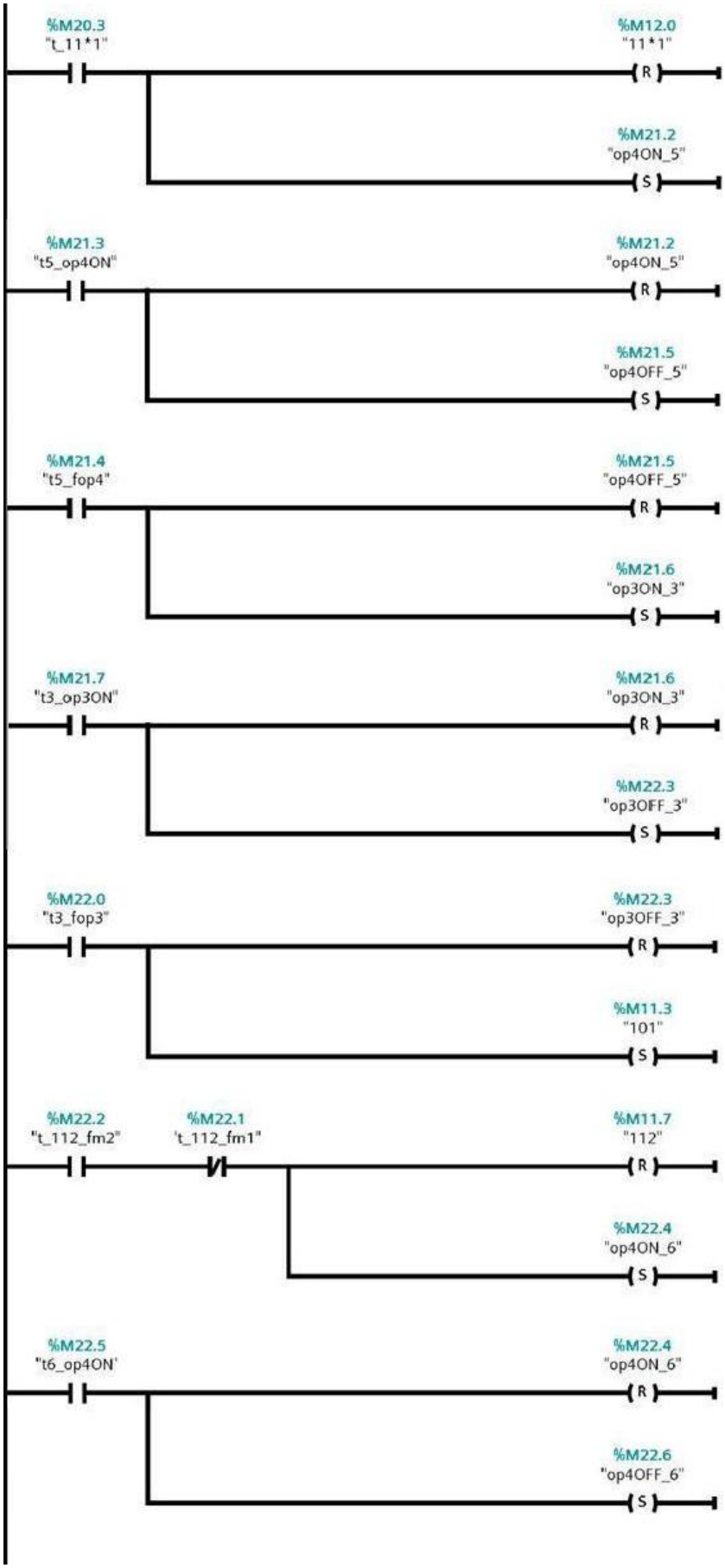

%M16.2
"t2_fop2"
%M15.7
"op2OFF_2"
R
%M11.2
"100"
S
%M17.0
"t_100"
%M11.2
"100"
R
%M17.1
"op4ON_1"
S
%M17.3
"t1_op4ON"
%M17.1
"op4ON_1"
R
%M17.2
"op4OFF_1"
S
%M17.4
"t1_fop4"
%M17.2
"op4OFF_1"
R
%M6.3
"000"
S
%M16.5
"t3_c2"
%M17.0
"t_100"
%M11.2
"100"
R
%M11.4
"102"
S
%M17.5
"t_102"
%M11.4
"102"
R
%M17.6
"op4ON_2"
S

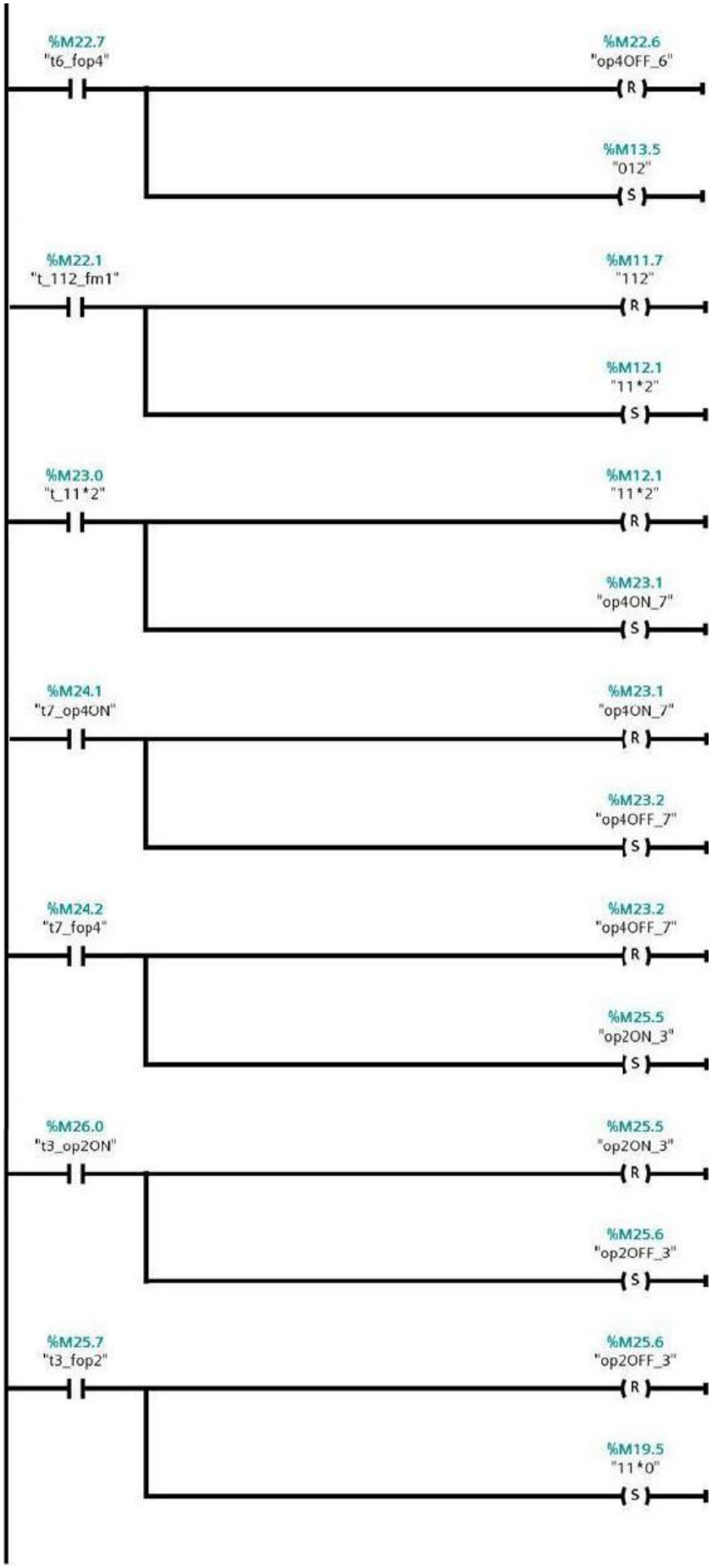

%M18.0
"t2_op4ON"
%M17.6
"op4ON_2"
R
%M17.7
"op4OFF_2"
S
%M18.1
"t2_fop4"
%M17.7
"op4OFF_2"
R
%M11.1
"Q02"
S
%M16.4
"t3_t1"
%M17.0
"t_100"
%M11.2
"100"
R
%M11.3
"101"
S
%M18.2
"t_101"
%M11.3
"101"
R
%M18.3
"op1ON_2"
S
%M23.7
"t2_op1ON"
%M18.3
"op1ON_2"
R
%M18.4
"op1OFF_2"
S
%M24.0
"t2_fop1"
%M18.4
"op1OFF_2"
R
%M11.5
"110"
S


%M18.5
"t4_c1"
%M19.3
"t_110_fm1"
%M19.4
"t_110_fm2"
%M11.5
"110"
R
%M11.6
"111"
S
%M18.6
"t4_c2"
%M19.3
"t_110_fm1"
%M19.4
"t_110_fm2"
%M11.5
"110"
R
%M11.7
"112"
S
%M19.3
"t_110_fm1"
%M11.5
"110"
R
%M19.5
"11*0"
S
%M19.4
"t_110_fm2"
%M19.3
"t_110_fm1"
%M11.5
"110"
R
%M19.6
"op4ON_3"
S
%M20.0
"t3_op4ON"
%M19.6
"op4ON_3"
R
%M19.7
"op4OFF_3"
S
%M20.1
"t3_fop4"
%M19.7
"op4OFF_3"
R
%M13.3
"010"
S

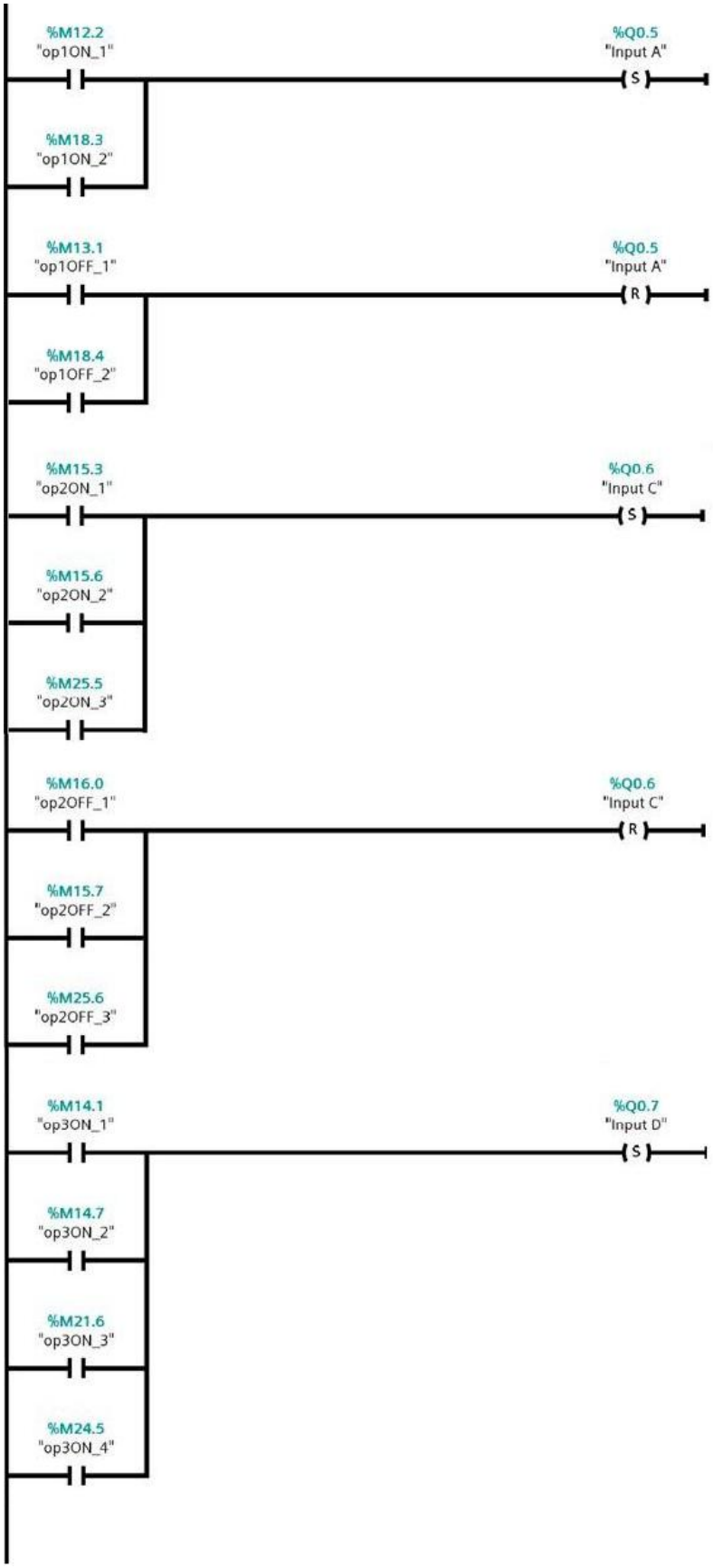

%M24.3
"t5_c1"
%M19.5
"11*0"
R
%M12.0
"11*1"
S
%M24.4
"t5_c2"
%M19.5
"11*0"
R
%M12.1
"11*2"
S
%M20.2
"t_11*0"
%M19.5
"11*0"
R
%M25.1
"op4ON_8"
S
%M25.3
"t8_op4ON"
%M25.1
"op4ON_8"
R
%M25.2
"op4OFF_8"
S
%M25.4
"t8_fop4"
%M6.0
"Ec2"
%M25.2
"op4OFF_8"
R
%M24.5
"op3ON_4"
S
%M6.0
"Ec2"
%M25.2
"op4OFF_8"
R
%M25.5
"op2ON_3"
S

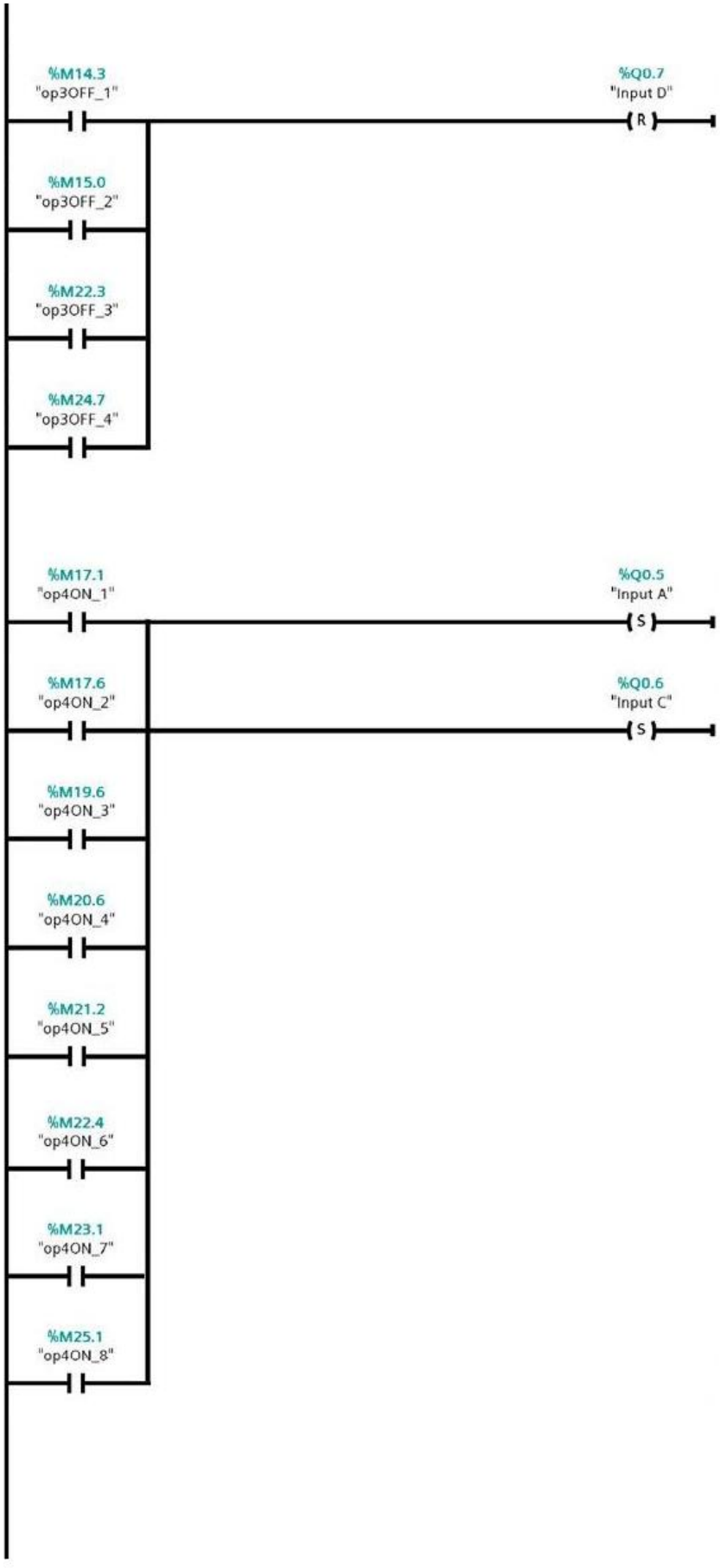

%M24.6
"t4_op3ON"
%M24.5
"op3ON_4"
R
%M24.7
"op3OFF_4"
S
%M25.0
"t4_fop3"
%M24.7
"op3OFF_4"
R
%M11.2
"100"
S
%M20.4
"t_111_fm1"
%M11.6
"111"
R
%M12.0
"11*1"
S
%M20.5
"t_111_fm2"
%M20.4
"t_111_fm1"
%M11.6
"111"
R
%M20.6
"op4ON_4"
S
%M20.7
"t4_op4ON"
%M20.6
"op4ON_4"
R
%M21.1
"op4OFF_4"
S
%M21.0
"t4_fop4"
%M21.1
"op4OFF_4"
R
%M13.4
"011"
S

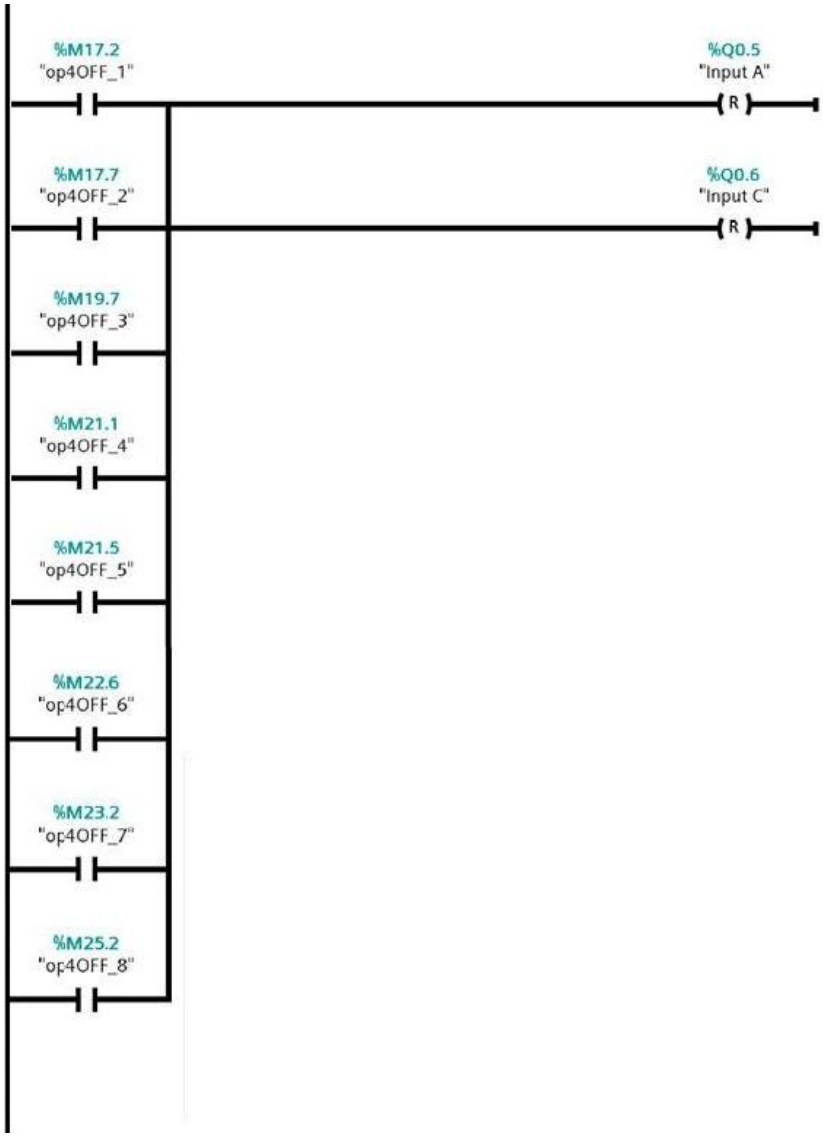

%M20.3
"t_11*1"
%M12.0
"11*1"
R
%M21.2
"op4ON_5"
S
%M21.3
"t5_op4ON"
%M21.2
"op4ON_5"
R
%M21.5
"op4OFF_5"
S
%M21.4
"t5_fop4"
%M21.5
"op4OFF_5"
R
%M21.6
"op3ON_3"
S
%M21.7
"t3_op3ON"
%M21.6
"op3ON_3"
R
%M22.3
"op3OFF_3"
S
%M22.0
"t3_fop3"
%M22.3
"op3OFF_3"
R
%M11.3
"101"
S
%M22.2
"t_112_fm2"
%M22.1
"t_112_fm1"
%M11.7
"112"
R
%M22.4
"op4ON_6"
S
%M22.5
"t6_op4ON"
%M22.4
"op4ON_6"
R
%M22.6
"op4OFF_6"
S


%M22.7
"t6_fop4"
%M22.6
"op4OFF_6"
R
%M13.5
"012"
S
%M22.1
"t_112_fm1"
%M11.7
"112"
R
%M12.1
"11*2"
S
%M23.0
"t_11*2"
%M12.1
"11*2"
R
%M23.1
"op4ON_7"
S
%M24.1
"t7_op4ON"
%M23.1
"op4ON_7"
R
%M23.2
"op4OFF_7"
S
%M24.2
"t7_fop4"
%M23.2
"op4OFF_7"
R
%M25.5
"op2ON_3"
S
%M26.0
"t3_op2ON"
%M25.5
"op2ON_3"
R
%M25.6
"op2OFF_3"
S
%M25.7
"t3_fop2"
%M25.6
"op2OFF_3"
R
%M19.5
"11*0"
S

Módulo de Ações




%M14.3
"op3OFF_1"
%Q0.7
"Input D"
R
%M15.0
"op3OFF_2"
%M22.3
"op3OFF_3"
%M24.7
"op3OFF_4"
%M17.1
"op4ON_1"
%Q0.5
"Input A"
S
%M17.6
"op4ON_2"
%Q0.6
"Input C"
S
%M19.6
"op4ON_3"
%M20.6
"op4ON_4"
%M21.2
"op4ON_5"
%M22.4
"op4ON_6"
%M23.1
"op4ON_7"
%M25.1
"op4ON_8"


%M17.2
"op4OFF_1"
%Q0.5
"Input A"
R
%M17.7
"op4OFF_2"
%Q0.6
"Input C"
R
%M19.7
"op4OFF_3"
%M21.1
"op4OFF_4"
%M21.5
"op4OFF_5"
%M22.6
"op4OFF_6"
%M23.2
"op4OFF_7"
%M25.2
"op4OFF_8"